Sync Smart Home Frontend Source Code

app/add-device/page.tsx

```
// add-device/pages.tsx
"use client"

import type React from "react"

import { useState } from "react"
import { useRouter } from "next/navigation"
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"
import { Button } from "@/components/ui/button"
import { Input } from "@/components/ui/input"
import { Label } from "@/components/ui/label"
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from
"@/components/ui/select"
import {
  Lightbulb,
  Thermometer,
  Fan,
  Tv,
  Lock,
  Plus,
  Plug,
  Coffee,
  Microwave,
  FolderIcon as Fridge,
  WashingMachine,
  ArrowLeft,
} from "lucide-react"
import { motion } from "framer-motion"

const deviceTypes = [
  { value: "light", label: "Light", icon: Lightbulb },
  { value: "thermostat", label: "Thermostat", icon: Thermometer },
  { value: "fan", label: "Fan", icon: Fan },
  { value: "tv", label: "TV", icon: Tv },
  { value: "lock", label: "Smart Lock", icon: Lock },
  { value: "plug", label: "Smart Plug", icon: Plug },
  { value: "coffee_maker", label: "Coffee Maker", icon: Coffee },
  { value: "microwave", label: "Microwave", icon: Microwave },
  { value: "refrigerator", label: "Refrigerator", icon: Fridge },
  { value: "washing_machine", label: "Washing Machine", icon: WashingMachine },
]

export default function AddDevicePage() {
  const router = useRouter()
  const [deviceName, setDeviceName] = useState("")
  const [deviceType, setDeviceType] = useState("")
```

```
  const [room, setRoom] = useState("")

  const handleAddDevice = (e: React.FormEvent) => {
    e.preventDefault()

    try{
      const newDevice = {
        id: Date.now().toString(),
        name: deviceName,
        type: deviceType,
        room: room,
        status: "off",
        powerConsumption: getDevicePowerConsumption(deviceType),
        lastStatusChange: Date.now(),
        totalEnergyConsumed: 0,
      }

      const existingDevices = JSON.parse(localStorage.getItem("devices") || "[]")
      const updatedDevices = [...existingDevices, newDevice]
      localStorage.setItem("devices", JSON.stringify(updatedDevices))

      router.push("/devices")
    } catch (error) {
      console.error("Failed to add device:", error)
    }
  }

  const getDevicePowerConsumption = (deviceType: string): number => {
    const consumptionMap: { [key: string]: number } = {
      light: 10,
      thermostat: 50,
      fan: 60,
      tv: 100,
      refrigerator: 150,
      washing_machine: 500,
      microwave: 1000,
    }
    return consumptionMap[deviceType] || 50
  }

  return (
    <div className="p-6 bg-gray-50 min-h-screen">
      <header className="flex justify-between items-center mb-6">
        <div className="flex items-center gap-2">
          <Button variant="ghost" size="icon" onClick={() => router.back()}>
            <ArrowLeft className="h-6 w-6" />
          </Button>
          <div>
            <h1 className="text-2xl font-bold text-gray-800">Add New Device</h1>
            <p className="text-sm text-gray-500">Connect a new smart device to your
home</p>
          </div>
```

```
        </div>
      </header>

      <Card className="max-w-2xl mx-auto">
        <CardHeader>
          <CardTitle className="text-[#00B2FF]">Device Information</CardTitle>
        </CardHeader>
        <CardContent>
          <form onSubmit={handleAddDevice} className="space-y-6">
            <div className="space-y-2">
              <Label htmlFor="name">Device Name</Label>
              <Input
                id="name"
                value={deviceName}
                onChange={(e) => setDeviceName(e.target.value)}
                placeholder="Enter device name"
                required
              />
            </div>
            <div className="space-y-2">
              <Label htmlFor="type">Device Type</Label>
              <Select value={deviceType} onValueChange={setDeviceType} required>
                <SelectTrigger className="w-full">
                  <SelectValue placeholder="Select device type" />
                </SelectTrigger>
                <SelectContent>
                  {deviceTypes.map((type) => (
                    <SelectItem key={type.value} value={type.value}>
                      <div className="flex items-center">
                        <type.icon className="mr-2 h-4 w-4 text-[#00B2FF]" />
                        {type.label}
                      </div>
                    </SelectItem>
                  ))}
                </SelectContent>
              </Select>
            </div>
            <div className="space-y-2">
              <Label htmlFor="room">Room</Label>
              <Select value={room} onValueChange={setRoom} required>
                <SelectTrigger className="w-full">
                  <SelectValue placeholder="Select room" />
                </SelectTrigger>
                <SelectContent>
                  {JSON.parse(localStorage.getItem("rooms") || "[]").map((room: any)
=> (
                    <SelectItem key={room.id} value={room.name}>
                      {room.name}
                    </SelectItem>
                  ))}
                </SelectContent>
              </Select>
```

```
            </div>
            <div className="flex gap-4">
              <Button type="button" variant="outline" className="flex-1" onClick={()
=> router.back()}>
                Cancel
              </Button>
              <Button type="submit" className="flex-1 bg-[#00B2FF] hover:bg-
[#00B2FF]/90">
                <Plus className="mr-2 h-4 w-4" /> Add Device
              </Button>
            </div>
          </form>
        </CardContent>
      </Card>

      <motion.div
        initial={{ opacity: 0, y: 20 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ delay: 0.2 }}
        className="mt-8 max-w-2xl mx-auto"
      >
        <h2 className="text-xl font-semibold mb-4">Compatible Devices</h2>
        <div className="grid grid-cols-2 sm:grid-cols-3 md:grid-cols-4 gap-4">
          {deviceTypes.map((type) => (
            <Card
              key={type.value}
              className="flex flex-col items-center p-4 cursor-pointer hover:shadow-
md transition-shadow"
            >
              <type.icon className="h-8 w-8 text-[#00B2FF] mb-2" />
              <p className="text-sm text-center">{type.label}</p>
            </Card>
          ))}
        </div>
      </motion.div>
    </div>
  )
}
```

app/add-room/page.tsx

```
// add-room/pages.tsx
"use client";

import type React from "react";

import { useState, useEffect } from "react";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";
import { Button } from "@/components/ui/button";
```

```tsx
import { Input } from "@/components/ui/input";
import { Label } from "@/components/ui/label";
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from
"@/components/ui/select";
import { User, Upload, Trash2, ArrowLeft, Home } from "lucide-react";
import { useRouter } from "next/navigation";
import Image from "next/image";
import { motion } from "framer-motion";
import { NavigationSidebar } from "@/app/components/navigation-sidebar"; // Import
the navbar

interface Room {
  id: number;
  name: string;
  type: string;
  image: string;
}

export default function AddRoomPage() {
  const router = useRouter();
  const [formData, setFormData] = useState({
    name: "",
    type: "",
    image: "",
  });
  const [rooms, setRooms] = useState<Room[]>([]);

  useEffect(() => {
    const storedRooms = JSON.parse(localStorage.getItem("rooms") || "[]");
    setRooms(storedRooms);
  }, []);

  const handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    const newRoom = {
      id: Date.now(),
      ...formData,
    };
    const updatedRooms = [...rooms, newRoom];
    localStorage.setItem("rooms", JSON.stringify(updatedRooms));
    setRooms(updatedRooms);
    setFormData({ name: "", type: "", image: "" });
  };

  const handleDelete = (id: number) => {
    const updatedRooms = rooms.filter((room) => room.id !== id);
    localStorage.setItem("rooms", JSON.stringify(updatedRooms));
    setRooms(updatedRooms);
  };

  const containerVariants = {
    hidden: { opacity: 0 },
```

```
    visible: {
      opacity: 1,
      transition: {
        staggerChildren: 0.1,
      },
    },
  };

  const itemVariants = {
    hidden: { y: 20, opacity: 0 },
    visible: {
      y: 0,
      opacity: 1,
    },
  };

  return (
    <div className="min-h-screen bg-gray-50 flex">
       <NavigationSidebar /> {/* Add the navbar here */}
      <div className="flex-1 ml-[72px]">
        <motion.div initial="hidden" animate="visible" variants={containerVariants}
className="p-6">
          <motion.header variants={itemVariants} className="flex justify-between
items-center mb-6">
            <div className="flex items-center gap-4">
              <div className="w-10 h-10 bg-gradient-to-br from-[#00B2FF] to-
[#0085FF] rounded-full flex items-center justify-center">
                <span className="text-white font-bold text-xl">Sy</span>
              </div>
              <div>
                <h1 className="text-2xl font-bold text-gray-800">Add New Room</h1>
                <p className="text-sm text-gray-500">Create a new room in your smart
home</p>
              </div>
            </div>
            <div className="flex items-center gap-2">
              <Button variant="ghost" size="icon" onClick={() => router.back()}>
                <ArrowLeft className="h-6 w-6" />
              </Button>
              <Button variant="ghost" size="icon">
                <User className="h-5 w-5" />
              </Button>
            </div>
          </motion.header>

          <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
            <motion.div variants={itemVariants}>
              <Card className="bg-white shadow-lg">
                <CardHeader>
                  <CardTitle className="text-2xl font-semibold text-[#00B2FF]">Room
Details</CardTitle>
                </CardHeader>
```

```
                <CardContent>
                  <form onSubmit={handleSubmit} className="space-y-6">
                    <div className="space-y-2">
                      <Label htmlFor="name" className="text-gray-700">
                        Room Name
                      </Label>
                      <Input
                        id="name"
                        placeholder="Enter room name"
                        value={formData.name}
                        onChange={(e) => setFormData({ ...formData, name:
e.target.value })}
                        required
                        className="border-[#00B2FF] focus:ring-[#00B2FF]"
                      />
                    </div>

                    <div className="space-y-2">
                      <Label htmlFor="type" className="text-gray-700">
                        Room Type
                      </Label>
                      <Select value={formData.type} onValueChange={(value) =>
setFormData({ ...formData, type: value })}>
                        <SelectTrigger className="border-[#00B2FF] focus:ring-
[#00B2FF]">
                          <SelectValue placeholder="Select room type" />
                        </SelectTrigger>
                        <SelectContent>
                          <SelectItem value="living">Living Room</SelectItem>
                          <SelectItem value="bedroom">Bedroom</SelectItem>
                          <SelectItem value="kitchen">Kitchen</SelectItem>
                          <SelectItem value="bathroom">Bathroom</SelectItem>
                          <SelectItem value="office">Office</SelectItem>
                          <SelectItem value="garage">Garage</SelectItem>
                        </SelectContent>
                      </Select>
                    </div>

                    <div className="space-y-2">
                      <Label className="text-gray-700">Room Image</Label>
                      <div className="border-2 border-dashed border-[#00B2FF]
rounded-lg p-8 text-center hover:bg-blue-50 transition-colors">
                        <Upload className="w-12 h-12 mx-auto mb-4 text-[#00B2FF]" />
                        <div className="space-y-2">
                          <p className="text-sm text-gray-600">Drag and drop an
image, or click to select</p>
                          <Input
                            type="file"
                            className="hidden"
                            accept="image/*"
                            onChange={(e) => {
                              const file = e.target.files?.[0];
```

```
                            if (file) {
                              const reader = new FileReader();
                              reader.onloadend = () => {
                                setFormData({ ...formData, image: reader.result as
string });
                              };
                              reader.readAsDataURL(file);
                            }
                          }}
                        />
                        <Button
                          type="button"
                          variant="outline"
                          onClick={() =>
document.querySelector('input[type="file"]')?.click()}
                          className="border-[#00B2FF] text-[#00B2FF] hover:bg-
[#00B2FF] hover:text-white"
                        >
                          Select Image
                        </Button>
                      </div>
                    </div>
                  </div>

                  <div className="flex gap-4">
                    <Button
                      type="button"
                      variant="outline"
                      className="flex-1 border-[#00B2FF] text-[#00B2FF] hover:bg-
[#00B2FF] hover:text-white"
                      onClick={() => router.push("/")}
                    >
                      Cancel
                    </Button>
                    <Button type="submit" className="flex-1 bg-[#00B2FF] hover:bg-
[#00B2FF]/90">
                      Add Room
                    </Button>
                  </div>
                </form>
              </CardContent>
            </Card>
          </motion.div>

          <motion.div variants={itemVariants} className="space-y-4">
            <h2 className="text-xl font-semibold text-gray-800">Current Rooms</h2>
            <motion.div variants={containerVariants} className="grid grid-cols-1
md:grid-cols-2 gap-4">
              {rooms.map((room) => (
                <motion.div key={room.id} variants={itemVariants}>
                  <Card className="overflow-hidden hover:shadow-lg transition-
shadow">
```

```
                    <div className="relative h-40">
                      <Image src={room.image || "/placeholder.svg"} alt=
{room.name} layout="fill" objectFit="cover" />
                    </div>
                    <CardContent className="p-4">
                      <div className="flex items-center justify-between">
                        <div>
                          <h3 className="font-medium text-lg text-gray-800">
{room.name}</h3>
                          <p className="text-sm text-gray-500 capitalize">
{room.type}</p>
                        </div>
                        <Button variant="ghost" size="icon" onClick={() =>
handleDelete(room.id)}>
                          <Trash2 className="h-5 w-5 text-red-500" />
                        </Button>
                      </div>
                    </CardContent>
                  </Card>
                </motion.div>
              ))}
            </motion.div>
            {rooms.length === 0 && (
              <motion.div variants={itemVariants}>
                <Card className="p-6 text-center">
                  <Home className="w-12 h-12 mx-auto mb-4 text-gray-400" />
                  <h3 className="text-lg font-semibold text-gray-700 mb-2">No
rooms added yet</h3>
                  <p className="text-gray-500">Start by adding your first room
above</p>
                </Card>
              </motion.div>
            )}
          </motion.div>
        </div>
      </motion.div>
    </div>
  );
}
```

app/auth/forgot-password/page.tsx

```
// forgot-password/page.tsx
"use client"

import type React from "react"

import { useState, useRef, useEffect } from "react"
import { useRouter } from "next/navigation"
```

```tsx
import { Button } from "@/components/ui/button"
import { Input } from "@/components/ui/input"
import { toast } from "@/components/ui/use-toast"
import { ArrowLeft, Mail, Lock } from "lucide-react"
import Link from "next/link"
import { Card, CardContent, CardDescription, CardFooter, CardHeader, CardTitle }
from "@/components/ui/card"

export default function ForgotPasswordPage() {
  const router = useRouter()
  const [step, setStep] = useState(1)
  const [email, setEmail] = useState("")
  const [otp, setOtp] = useState(["", "", "", "", "", ""])
  const [newPassword, setNewPassword] = useState("")
  const [confirmPassword, setConfirmPassword] = useState("")
  const otpRefs = useRef<(HTMLInputElement | null)[]>([])

  useEffect(() => {
    otpRefs.current = otpRefs.current.slice(0, 6)
  }, [])

  const handleSendOTP = (e: React.FormEvent) => {
    e.preventDefault()
    toast({
      title: "OTP Sent",
      description: "Please check your email for the OTP.",
    })
    setStep(2)
  }

  const handleVerifyOTP = (e: React.FormEvent) => {
    e.preventDefault()
    if (otp.join("") === "123456") {
      setStep(3)
    } else {
      toast({
        title: "Invalid OTP",
        description: "Please enter the correct OTP.",
        variant: "destructive",
      })
    }
  }

  const handleResetPassword = (e: React.FormEvent) => {
    e.preventDefault()
    if (newPassword !== confirmPassword) {
      toast({
        title: "Passwords do not match",
        description: "Please make sure your passwords match.",
        variant: "destructive",
      })
      return
```

```
    }
    toast({
      title: "Password Reset Successful",
      description: "Your password has been reset. Please log in with your new
password.",
    })
    router.push("/auth/login")
  }

  const handleOtpChange = (index: number, value: string) => {
    if (value.length > 1) return
    const newOtp = [...otp]
    newOtp[index] = value
    setOtp(newOtp)
    if (value && index < 5) {
      otpRefs.current[index + 1]?.focus()
    }
  }

  const handleOtpKeyDown = (index: number, e: React.KeyboardEvent<HTMLInputElement>)
=> {
    if (e.key === "Backspace" && !otp[index] && index > 0) {
      otpRefs.current[index - 1]?.focus()
    }
  }

  return (
    <div className="min-h-screen flex items-center justify-center bg-gray-50 py-12
px-4 sm:px-6 lg:px-8">
      <Card className="w-full max-w-md">
        <CardHeader>
          <div className="flex justify-between items-center mb-4">
            <Link href="/auth/login" className="text-[#00B2FF] hover:underline flex
items-center">
              <ArrowLeft className="mr-2 h-4 w-4" />
              Back to Login
            </Link>
            <div className="w-10 h-10 bg-[#00B2FF] rounded-full flex items-center
justify-center">
              <span className="text-white font-bold text-xl">Sy</span>
            </div>
          </div>
          <CardTitle className="text-2xl font-bold text-center text-gray-900">Reset
your password</CardTitle>
          <CardDescription className="text-center text-gray-500">
            {step === 1 && "Enter your email to receive a one-time password"}
            {step === 2 && "Enter the OTP sent to your email"}
            {step === 3 && "Create a new password for your account"}
          </CardDescription>
        </CardHeader>
        <CardContent>
          {step === 1 && (
```

```
            <form onSubmit={handleSendOTP} className="space-y-4">
              <div className="relative">
                <Mail className="absolute left-3 top-1/2 -translate-y-1/2 text-gray-
400" />
                <Input
                  type="email"
                  placeholder="Enter your email"
                  value={email}
                  onChange={(e) => setEmail(e.target.value)}
                  className="pl-10 border-gray-300 focus:border-[#00B2FF]
focus:ring-[#00B2FF]"
                  required
                />
              </div>
              <Button type="submit" className="w-full bg-[#FF9500] hover:bg-
[#FF9500]/90 text-white">
                Send OTP
              </Button>
            </form>
          )}
          {step === 2 && (
            <form onSubmit={handleVerifyOTP} className="space-y-4">
              <div className="flex justify-between gap-2">
                {otp.map((digit, index) => (
                  <Input
                    key={index}
                    type="text"
                    inputMode="numeric"
                    pattern="[0-9]*"
                    maxLength={1}
                    className="w-12 h-12 text-center text-2xl border-gray-300
focus:border-[#00B2FF] focus:ring-[#00B2FF]"
                    value={digit}
                    onChange={(e) => handleOtpChange(index, e.target.value)}
                    onKeyDown={(e) => handleOtpKeyDown(index, e)}
                    ref={(el) => (otpRefs.current[index] = el)}
                  />
                ))}
              </div>
              <Button type="submit" className="w-full bg-[#FF9500] hover:bg-
[#FF9500]/90 text-white">
                Verify OTP
              </Button>
            </form>
          )}
          {step === 3 && (
            <form onSubmit={handleResetPassword} className="space-y-4">
              <div className="relative">
                <Lock className="absolute left-3 top-1/2 -translate-y-1/2 text-gray-
400" />
                <Input
                  type="password"
```

```
                placeholder="New Password"
                value={newPassword}
                onChange={(e) => setNewPassword(e.target.value)}
                className="pl-10 border-gray-300 focus:border-[#00B2FF]
focus:ring-[#00B2FF]"
                required
              />
            </div>
            <div className="relative">
              <Lock className="absolute left-3 top-1/2 -translate-y-1/2 text-gray-
400" />
              <Input
                type="password"
                placeholder="Confirm New Password"
                value={confirmPassword}
                onChange={(e) => setConfirmPassword(e.target.value)}
                className="pl-10 border-gray-300 focus:border-[#00B2FF]
focus:ring-[#00B2FF]"
                required
              />
            </div>
            <Button type="submit" className="w-full bg-[#FF9500] hover:bg-
[#FF9500]/90 text-white">
              Reset Password
            </Button>
          </form>
        )}
      </CardContent>
      <CardFooter className="text-center">
        <p className="text-sm text-gray-600">
          Remember your password?{" "}
          <Link href="/auth/login" className="text-[#00B2FF] hover:underline">
            Login here
          </Link>
        </p>
      </CardFooter>
    </Card>
  </div>
 )
}
```

app/auth/household-login/page.tsx

```
// household-login/page.tsx
"use client"

import type React from "react"
import Image from "next/image"
import { useState } from "react"
```

```tsx
import { useRouter } from "next/navigation"
import { Button } from "@/components/ui/button"
import { Input } from "@/components/ui/input"
import Link from "next/link"
import { Mail } from "lucide-react"
import { toast } from "@/components/ui/use-toast"

export default function HouseholdLoginPage() {
  const router = useRouter()
  const [formData, setFormData] = useState({
    email: "",
    pin: ["", "", "", ""],
  })

  const handlePinChange = (index: number, value: string) => {
    if (value.length > 1) return
    const newPin = [...formData.pin]
    newPin[index] = value
    setFormData({ ...formData, pin: newPin })

    if (value && index < 3) {
      const nextInput = document.getElementById(`pin-${index + 1}`)
      nextInput?.focus()
    }
  }

  const handleSubmit = (e: React.FormEvent) => {
    e.preventDefault()
    const familyMembers = JSON.parse(localStorage.getItem("familyMembers") || "[]")
    const member = familyMembers.find((m: any) => m.email === formData.email &&
m.pin === formData.pin.join(""))

    if (member) {
      localStorage.setItem(
        "currentUser",
        JSON.stringify({
          id: member.id,
          type: "household",
          email: member.email,
          name: member.name,
          role: member.role,
          permissions: member.permissions,
        }),
      )
      router.push("/")
    } else {
      toast({
        title: "Login Failed",
        description: "Invalid email or PIN. Please try again.",
        variant: "destructive",
      })
    }
```

```
  }

  return (
    <div className="min-h-screen flex">
      <div className="hidden lg:flex lg:flex-1 relative">
        <Image
          src="https://media.istockphoto.com/id/1219569858/photo/woman-cooking-on-
the-modern-kitchen-at-home.jpg?
s=612x612&w=0&k=20&c=oUTIUAb0_cALWZ2GCTI3ZUmZCH31cf0UmBMPf1tMsck="
          alt="Smart Home"
          fill
          className="object-cover"
        />
        <div className="absolute inset-0 bg-black/20" />
        <div className="absolute bottom-20 left-10 text-white">
          <h1 className="text-5xl font-bold mb-4">
            <span className="text-[#00B2FF]">SYNC</span> your Home,
            <br />
            Save energy,
            <br />
            and live smarter.
          </h1>
        </div>
      </div>

      <div className="flex-1 flex items-center justify-center p-8">
        <div className="w-full max-w-md">
          <div className="flex justify-end mb-8">
            <div className="flex items-center gap-2">
              <span className="text-2xl font-bold bg-[#00B2FF] text-white px-3 py-1
rounded-full">
                Sy<span className="text-[#FFB800]">nc</span>
              </span>
            </div>
          </div>

          <div className="mb-8">
            <h2 className="text-2xl font-semibold">Welcome,</h2>
            <p className="text-gray-600">Household Member</p>
          </div>

          <form onSubmit={handleSubmit} className="space-y-6">
            <div className="space-y-2">
              <label className="text-sm font-medium">Enter your Email</label>
              <div className="relative">
                <Input
                  type="email"
                  value={formData.email}
                  onChange={(e) => setFormData({ ...formData, email: e.target.value
})}
                  className="pl-10"
                  placeholder="Enter your Email"
```

```
                required
              />
              <Mail className="absolute left-3 top-1/2 -translate-y-1/2 h-5 w-5
text-gray-400" />
            </div>
          </div>

          <div className="space-y-2">
            <label className="text-sm font-medium">PIN</label>
            <div className="flex justify-between gap-4">
              {[0, 1, 2, 3].map((i) => (
                <Input
                  key={i}
                  id={`pin-${i}`}
                  type="text"
                  inputMode="numeric"
                  maxLength={1}
                  value={formData.pin[i]}
                  onChange={(e) => handlePinChange(i, e.target.value)}
                  className="w-16 h-16 text-center text-2xl"
                  required
                />
              ))}
            </div>
          </div>

          <Button type="submit" className="w-full bg-[#FF9500] hover:bg-
[#FF9500]/90 py-6 text-lg">
            Login
          </Button>

          <div className="text-center">
            <Link href="/auth/login" className="text-[#00B2FF] hover:underline
text-sm">
              ← Back to Admin Login
            </Link>
          </div>
        </form>
      </div>
    </div>
  </div>
  )
}
```

app/auth/login/page.tsx

```
// login/page.tsx
"use client"
```

```tsx
import Image from "next/image"
import { useState } from "react";
import { useRouter } from "next/navigation";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import Link from "next/link";
import { Eye, EyeOff, Mail } from "lucide-react";
import { toast } from "@/components/ui/use-toast";
import axios from 'axios';
import { useUser } from "@/contexts/UserContext"; // Import the user context

// API base URL with fallback
const API_URL = process.env.NEXT_PUBLIC_API_URL || 'http://localhost:8000/api/v1';

export default function LoginPage() {
  const router = useRouter();
  const { login } = useUser(); // Use the login function from context
  const [showPassword, setShowPassword] = useState(false);
  const [formData, setFormData] = useState({ email: "", password: "" });
  const [isLoading, setIsLoading] = useState(false);
  const [isForgotPassword, setIsForgotPassword] = useState(false);
  const [otp, setOtp] = useState("");
  const [newPassword, setNewPassword] = useState("");
  const [verificationEmail, setVerificationEmail] = useState("");

  // Handle login form submission
  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();

    // Validate inputs
    if (!formData.email || !formData.password) {
      toast({
        title: "Missing Information",
        description: "Please enter both email and password",
        variant: "destructive",
      });
      return;
    }

    setIsLoading(true);

    try {
      // Using the correct API endpoint format from the documentation
      const response = await axios.post(`${API_URL}/users/token`,
        new URLSearchParams({
          'username': formData.email, // The API expects 'username' for email
          'password': formData.password
        }),
        {
          headers: {
            'Content-Type': 'application/x-www-form-urlencoded'
          }
```

```
      }
    );

    // Extract token and store it
    const { access_token, token_type } = response.data;

    // Use the login function from context to update global state
    login({
      email: formData.email,
      isAuthenticated: true,
      token: access_token
    });

    toast({
      title: "Login Successful",
      description: "Welcome back!",
      variant: "default",
    });

    // Redirect to dashboard
    router.push("/dashboard");
  } catch (error) {
    // Handle common error scenarios
    console.error("Login error:", error);

    const errorMessage =
      (error as any).response?.data?.detail ||
      (error as any).message ||
      "Authentication failed. Please check your credentials and try again.";

    toast({
      title: "Login Failed",
      description: errorMessage,
      variant: "destructive",
    });
  } finally {
    setIsLoading(false);
  }
};

// Handle forgot password request
const handleForgotPassword = async () => {
  if (!formData.email) {
    toast({
      title: "Email Required",
      description: "Please enter your email address",
      variant: "destructive",
    });
    return;
  }

  setIsLoading(true);
```

```
    try {
      // Store the email for the verification step
      setVerificationEmail(formData.email);

      // Request password reset
      await axios.post(`${API_URL}/users/request-password-reset`, {
        email: formData.email
      });

      toast({
        title: "Verification Email Sent",
        description: "A confirmation link has been sent to your email",
        variant: "default",
      });

      setIsForgotPassword(true);
    } catch (error) {
      const errorMessage =
        (error as any).response?.data?.detail ||
        (error as any).message ||
        "Failed to process your request. The email may not be registered.";

      toast({
        title: "Request Failed",
        description: errorMessage,
        variant: "destructive",
      });
    } finally {
      setIsLoading(false);
    }
  };

  // Handle password reset
  const handleResetPassword = async (e: React.FormEvent) => {
    e.preventDefault();

    if (!otp || !newPassword) {
      toast({
        title: "Missing Information",
        description: "Please enter both the verification code and new password",
        variant: "destructive",
      });
      return;
    }

    // Validate password
    const passwordRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[!@#$%^&*()_+\-=\[\]
{};':",.<>?/])[A-Za-z\d!@#$%^&*()_+\-=\[\]{};':",.<>?/]{8,}$/;
    if (!passwordRegex.test(newPassword)) {
      toast({
        title: "Invalid Password",
```

```
        description: "Password must be at least 8 characters with uppercase,
lowercase, numbers, and special characters.",
        variant: "destructive",
      });
      return;
    }

    setIsLoading(true);

    try {
      // Verify the OTP
      await axios.post(`${API_URL}/users/verify-reset-code`, {
        email: verificationEmail || formData.email,
        code: otp
      });

      // Update the password
      await axios.post(`${API_URL}/users/reset-password`, {
        email: verificationEmail || formData.email,
        new_password: newPassword
      });

      toast({
        title: "Password Reset Successful",
        description: "Your password has been updated. You can now log in with your
new password.",
        variant: "default",
      });

      // Reset form state
      setIsForgotPassword(false);
      setOtp("");
      setNewPassword("");
      setFormData({ ...formData, password: "" });
    } catch (error) {
      const errorMessage =
        (error as any).response?.data?.detail ||
        (error as any).message ||
        "Failed to reset password. Please try again or request a new code.";

      toast({
        title: "Reset Failed",
        description: errorMessage,
        variant: "destructive",
      });
    } finally {
      setIsLoading(false);
    }
  };

  return (
    <div className="min-h-screen flex">
```

```
      <div className="hidden lg:flex lg:flex-1 relative">
        <Image
          src="https://media.istockphoto.com/id/1219569858/photo/woman-cooking-on-
the-modern-kitchen-at-home.jpg?
s=612x612&w=0&k=20&c=oUTIUAb0_cALWZ2GCTI3ZUmZCH31cf0UmBMPf1tMsck="
          alt="Smart Home"
          fill
          className="object-cover"
        />
        <div className="absolute inset-0 bg-black/20" />
        <div className="absolute bottom-20 left-10 text-white">
          <h1 className="text-5xl font-bold mb-4">
            <span className="text-[#00B2FF]">SYNC</span> your Home,
            <br />
            Save energy,
            <br />
            and live smarter.
          </h1>
          <Link href="/auth/signup">
            <Button className="bg-[#FF9500] hover:bg-[#FF9500]/90 text-lg px-8 py-
6">
              Sign Up
            </Button>
          </Link>
        </div>
      </div>

      <div className="flex-1 flex items-center justify-center p-8">
        <div className="w-full max-w-md">
          <div className="flex justify-end mb-8">
            <div className="flex items-center gap-2">
              <span className="text-2xl font-bold bg-[#00B2FF] text-white px-3 py-1
rounded-full">
                Sy<span className="text-[#FFB800]">nc</span>
              </span>
            </div>
          </div>

          <div className="mb-8">
            <h2 className="text-2xl font-semibold">Welcome,</h2>
            <p className="text-gray-600">Smart Home Login</p>
          </div>

          {!isForgotPassword ? (
            <form onSubmit={handleSubmit} className="space-y-6">
              <div className="space-y-2">
                <label className="text-sm font-medium">Email</label>
                <div className="relative">
                  <Input
                    type="email"
                    value={formData.email}
                    onChange={(e) => setFormData({ ...formData, email:
```

```
e.target.value })}
                    className="pl-10"
                    placeholder="Enter your email"
                    required
                  />
                  <Mail className="absolute left-3 top-1/2 -translate-y-1/2 h-5 w-5
text-gray-400" />
                </div>
              </div>

              <div className="space-y-2">
                <label className="text-sm font-medium">Password</label>
                <div className="relative">
                  <Input
                    type={showPassword ? "text" : "password"}
                    value={formData.password}
                    onChange={(e) => setFormData({ ...formData, password:
e.target.value })}
                    className="pr-10"
                    placeholder="Enter your password"
                    required
                  />
                  <button
                    type="button"
                    onClick={() => setShowPassword(!showPassword)}
                    className="absolute right-3 top-1/2 -translate-y-1/2"
                  >
                    {showPassword ? (
                      <EyeOff className="h-5 w-5 text-gray-400" />
                    ) : (
                      <Eye className="h-5 w-5 text-gray-400" />
                    )}
                  </button>
                </div>
              </div>

              <div className="flex justify-between items-center">
                <button
                  type="button"
                  onClick={handleForgotPassword}
                  className="text-sm text-[#00B2FF] hover:underline"
                >
                  Forgot Password?
                </button>
              </div>

              <Button
                type="submit"
                className="w-full bg-[#FF9500] hover:bg-[#FF9500]/90 py-6 text-lg"
                disabled={isLoading}
              >
                {isLoading ? 'Logging In...' : 'Login'}
```

```
              </Button>

              <div className="text-center space-y-2">
                <p className="text-sm text-gray-600">
                  {"Don't have an account? "}
                  <Link href="/auth/signup" className="text-[#00B2FF] bg-blue-50 px-
3 py-1 rounded-full text-sm hover:bg-blue-100">
                    Register Now
                  </Link>
                </p>
                <div className="flex items-center gap-2 justify-center">
                  <p className="text-sm text-gray-600">Part of a Household?</p>
                  <Link
                    href="/auth/household-login"
                    className="text-[#00B2FF] bg-blue-50 px-3 py-1 rounded-full
text-sm hover:bg-blue-100"
                  >
                    Click Here
                  </Link>
                </div>
              </div>
            </form>
          ) : (
            <form onSubmit={handleResetPassword} className="space-y-6">
              <div className="space-y-2">
                <label className="text-sm font-medium">Verification Code</label>
                <Input
                  type="text"
                  value={otp}
                  onChange={(e) => setOtp(e.target.value)}
                  placeholder="Enter the verification code"
                  required
                />
                <p className="text-xs text-gray-500">
                  Enter the code sent to your email
                </p>
              </div>

              <div className="space-y-2">
                <label className="text-sm font-medium">New Password</label>
                <div className="relative">
                  <Input
                    type={showPassword ? "text" : "password"}
                    value={newPassword}
                    onChange={(e) => setNewPassword(e.target.value)}
                    placeholder="Enter your new password"
                    required
                  />
                  <button
                    type="button"
                    onClick={() => setShowPassword(!showPassword)}
                    className="absolute right-3 top-1/2 -translate-y-1/2"
```

```
                  >
                    {showPassword ? (
                      <EyeOff className="h-5 w-5 text-gray-400" />
                    ) : (
                      <Eye className="h-5 w-5 text-gray-400" />
                    )}
                  </button>
                </div>
                <p className="text-xs text-gray-500">
                  Password must be at least 8 characters with letters, numbers, and
special characters.
                </p>
              </div>

              <div className="flex space-x-2">
                <Button
                  type="button"
                  className="flex-1 bg-gray-200 hover:bg-gray-300 text-gray-800"
                  onClick={() => setIsForgotPassword(false)}
                >
                  Back to Login
                </Button>
                <Button
                  type="submit"
                  className="flex-1 bg-[#FF9500] hover:bg-[#FF9500]/90"
                  disabled={isLoading}
                >
                  {isLoading ? 'Resetting...' : 'Reset Password'}
                </Button>
              </div>
            </form>
          )}
        </div>
      </div>
    </div>
  );
}
```

app/auth/login/page.tsx.bak

```
// login/page.tsx
"use client"

import Image from "next/image"
import { useState } from "react";
import { useRouter } from "next/navigation";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import Link from "next/link";
import { Eye, EyeOff, Mail } from "lucide-react";
```

```tsx
import { toast } from "@/components/ui/use-toast";
import axios from 'axios';

const API_URL = process.env.NEXT_PUBLIC_API_URL || 'http://localhost:8000';

export default function LoginPage() {
  const router = useRouter();
  const [showPassword, setShowPassword] = useState(false);
  const [formData, setFormData] = useState({ email: "", password: "" });
  const [isLoading, setIsLoading] = useState(false);
  const [isForgotPassword, setIsForgotPassword] = useState(false);
  const [otp, setOtp] = useState("");
  const [newPassword, setNewPassword] = useState("");

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

    if (!formData.email || !formData.password) {
      toast({
        title: "Invalid Input",
        description: "Please fill in all fields",
        variant: "destructive",
      });
      return;
    }

    setIsLoading(true);

    try {
      const response = await axios.post(`${API_URL}/login`, formData);
      const userData = response.data;

      localStorage.setItem(
        "currentUser",
        JSON.stringify({
          id: userData.user_id,
          type: "admin",
          email: userData.admin_email,
          name: userData.firstName || "Admin",
          role: "Admin",
          householdId: userData.household_id,
        })
      );

      router.push("/dashboard"); // Update with your dashboard path
    } catch (error) {
      const errorMessage =
        (error as any).response?.data?.detail ||
        (error as any).message ||
        "An error occurred. Please try again.";
      toast({
        title: "Login Failed",
```

```
        description: errorMessage,
        variant: "destructive",
      });
    } finally {
      setIsLoading(false);
    }
  };

  const handleForgotPassword = async () => {
    if (!formData.email) {
      toast({
        title: "Invalid Input",
        description: "Please enter your email",
        variant: "destructive",
      });
      return;
    }

    try {
      await axios.post(`${API_URL}/request-forgot-password-otp`, { email:
formData.email });
      toast({
        title: "OTP Sent",
        description: "An OTP has been sent to your email",
        variant: "success",
      });
      setIsForgotPassword(true);
    } catch (error) {
      const errorMessage =
        (error as any).response?.data?.detail ||
        (error as any).message ||
        "An error occurred. Please try again.";
      toast({
        title: "Request Failed",
        description: errorMessage,
        variant: "destructive",
      });
    }
  };

  const handleResetPassword = async (e: React.FormEvent) => {
    e.preventDefault();

    if (!otp || !newPassword) {
      toast({
        title: "Invalid Input",
        description: "Please fill in all fields",
        variant: "destructive",
      });
      return;
    }
```

```
    try {
      // Verify OTP
      await axios.post(`${API_URL}/verify-forgot-password-otp`, { email:
formData.email, otp });

      // Reset Password
      await axios.post(`${API_URL}/reset-password`, { email: formData.email,
password: newPassword });
      toast({
        title: "Password Reset",
        description: "Your password has been reset successfully",
        variant: "success",
      });
      setIsForgotPassword(false);
    } catch (error) {
      const errorMessage =
        (error as any).response?.data?.detail ||
        (error as any).message ||
        "An error occurred. Please try again.";
      toast({
        title: "Reset Failed",
        description: errorMessage,
        variant: "destructive",
      });
    }
  };

  return (
    <div className="min-h-screen flex">
      <div className="hidden lg:flex lg:flex-1 relative">
        <Image
          src="https://media.istockphoto.com/id/1219569858/photo/woman-cooking-on-
the-modern-kitchen-at-home.jpg?
s=612x612&w=0&k=20&c=oUTIUAb0_cALWZ2GCTI3ZUmZCH31cf0UmBMPf1tMsck="
          alt="Smart Home"
          fill
          className="object-cover"
        />
        <div className="absolute inset-0 bg-black/20" />
        <div className="absolute bottom-20 left-10 text-white">
          <h1 className="text-5xl font-bold mb-4">
            <span className="text-[#00B2FF]">SYNC</span> your Home,
            <br />
            Save energy,
            <br />
            and live smarter.
          </h1>
          <Link href="/auth/signup">
            <Button className="bg-[#FF9500] hover:bg-[#FF9500]/90 text-lg px-8 py-
6">
              Sign Up
            </Button>
```

```
          </Link>
        </div>
      </div>

      <div className="flex-1 flex items-center justify-center p-8">
        <div className="w-full max-w-md">
          <div className="flex justify-end mb-8">
            <div className="flex items-center gap-2">
              <span className="text-2xl font-bold bg-[#00B2FF] text-white px-3 py-1
rounded-full">
                Sy<span className="text-[#FFB800]">nc</span>
              </span>
            </div>
          </div>

          <div className="mb-8">
            <h2 className="text-2xl font-semibold">Welcome,</h2>
            <p className="text-gray-600">Admin Login</p>
          </div>

          {!isForgotPassword ? (
            <form onSubmit={handleSubmit} className="space-y-6">
              <div className="space-y-2">
                <label className="text-sm font-medium">Email</label>
                <div className="relative">
                  <Input
                    type="email"
                    value={formData.email}
                    onChange={(e) => setFormData({ ...formData, email:
e.target.value })}
                    className="pl-10"
                    placeholder="Enter your email"
                    required
                  />
                  <Mail className="absolute left-3 top-1/2 -translate-y-1/2 h-5 w-5
text-gray-400" />
                </div>
              </div>

              <div className="space-y-2">
                <label className="text-sm font-medium">Password</label>
                <div className="relative">
                  <Input
                    type={showPassword ? "text" : "password"}
                    value={formData.password}
                    onChange={(e) => setFormData({ ...formData, password:
e.target.value })}
                    className="pr-10"
                    placeholder="Enter your password"
                    required
                  />
                  <button
```

```
                  type="button"
                  onClick={() => setShowPassword(!showPassword)}
                  className="absolute right-3 top-1/2 -translate-y-1/2"
                >
                  {showPassword ? (
                    <EyeOff className="h-5 w-5 text-gray-400" />
                  ) : (
                    <Eye className="h-5 w-5 text-gray-400" />
                  )}
                </button>
              </div>
            </div>

            <div className="flex justify-between items-center">
              <button
                type="button"
                onClick={handleForgotPassword}
                className="text-sm text-[#00B2FF] hover:underline"
              >
                Forgot Password?
              </button>
            </div>

            <Button
              type="submit"
              className="w-full bg-[#FF9500] hover:bg-[#FF9500]/90 py-6 text-lg"
              disabled={isLoading}
            >
              {isLoading ? 'Logging In...' : 'Login'}
            </Button>

            <div className="text-center space-y-2">
              <p className="text-sm text-gray-600">
                {"Don't have an account? "}
                <Link href="/auth/signup" className="text-[#00B2FF] bg-blue-50 px-
3 py-1 rounded-full text-sm hover:bg-blue-100">
                  Register Now
                </Link>
              </p>
              <div className="flex items-center gap-2 justify-center">
                <p className="text-sm text-gray-600">Part of a Household?</p>
                <Link
                  href="/auth/household-login"
                  className="text-[#00B2FF] bg-blue-50 px-3 py-1 rounded-full
text-sm hover:bg-blue-100"
                >
                  Click Here
                </Link>
              </div>
            </div>
          </form>
        ) : (
```

```
            <form onSubmit={handleResetPassword} className="space-y-6">
              <div className="space-y-2">
                <label className="text-sm font-medium">OTP</label>
                <Input
                  type="text"
                  value={otp}
                  onChange={(e) => setOtp(e.target.value)}
                  placeholder="Enter the OTP"
                  required
                />
              </div>

              <div className="space-y-2">
                <label className="text-sm font-medium">New Password</label>
                <Input
                  type="password"
                  value={newPassword}
                  onChange={(e) => setNewPassword(e.target.value)}
                  placeholder="Enter your new password"
                  required
                />
              </div>

              <Button
                type="submit"
                className="w-full bg-[#FF9500] hover:bg-[#FF9500]/90 py-6 text-lg"
                disabled={isLoading}
              >
                {isLoading ? 'Resetting Password...' : 'Reset Password'}
              </Button>
            </form>
          )}
        </div>
      </div>
    </div>
  );
}
```

app/auth/login/page.tsx.bak.bak

```
// login/page.tsx
"use client"

import Image from "next/image"
import { useState } from "react";
import { useRouter } from "next/navigation";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import Link from "next/link";
import { Eye, EyeOff, Mail } from "lucide-react";
```

```
import { toast } from "@/components/ui/use-toast";
import axios from 'axios';

// API base URL with fallback to localhost
const API_URL = process.env.NEXT_PUBLIC_API_URL || 'http://localhost:8000/api/v1';

export default function LoginPage() {
  const router = useRouter();
  const [showPassword, setShowPassword] = useState(false);
  const [formData, setFormData] = useState({ email: "", password: "" });
  const [isLoading, setIsLoading] = useState(false);
  const [isForgotPassword, setIsForgotPassword] = useState(false);
  const [otp, setOtp] = useState("");
  const [newPassword, setNewPassword] = useState("");
  const [verificationEmail, setVerificationEmail] = useState("");

  // Handle login form submission
  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();

    // Validate inputs
    if (!formData.email || !formData.password) {
      toast({
        title: "Missing Information",
        description: "Please enter both email and password",
        variant: "destructive",
      });
      return;
    }

    setIsLoading(true);

    try {
      // Using the correct API endpoint format from the documentation
      const response = await axios.post(`${API_URL}/users/token`,
        new URLSearchParams({
          'username': formData.email, // The API expects 'username' for email
          'password': formData.password
        }),
        {
          headers: {
            'Content-Type': 'application/x-www-form-urlencoded'
          }
        }
      );

      // Extract token and store it
      const { access_token, token_type } = response.data;

      // Get user details using the token
      const userResponse = await axios.get(`${API_URL}/users/me`, {
        headers: {
```

```
          'Authorization': `${token_type} ${access_token}`
        }
      });

      // Store auth token and user information
      localStorage.setItem('access_token', access_token);
      localStorage.setItem('token_type', token_type);
      localStorage.setItem('currentUser', JSON.stringify({
        email: formData.email,
        role: userResponse.data.role || 'user',
        id: userResponse.data.id,
        is_verified: userResponse.data.is_verified
      }));

      // Redirect to dashboard
      toast({
        title: "Login Successful",
        description: "Welcome back!",
        variant: "default",
      });

      router.push("/dashboard");
    } catch (error) {
      // Handle common error scenarios
      const errorMessage =
        (error as any).response?.data?.detail ||
        (error as any).message ||
        "Authentication failed. Please check your credentials and try again.";

      toast({
        title: "Login Failed",
        description: errorMessage,
        variant: "destructive",
      });
    } finally {
      setIsLoading(false);
    }
  };

  // Handle forgot password request
  const handleForgotPassword = async () => {
    if (!formData.email) {
      toast({
        title: "Email Required",
        description: "Please enter your email address",
        variant: "destructive",
      });
      return;
    }

    setIsLoading(true);
```

```
    try {
      // Store the email for the verification step
      setVerificationEmail(formData.email);

      // Using the documented password reset flow
      await axios.post(`${API_URL}/users/request-password-reset`, {
        email: formData.email
      });

      toast({
        title: "Verification Code Sent",
        description: "Please check your email for the verification code",
        variant: "default",
      });

      setIsForgotPassword(true);
    } catch (error) {
      const errorMessage =
        (error as any).response?.data?.detail ||
        (error as any).message ||
        "Failed to process your request. The email may not be registered.";

      toast({
        title: "Request Failed",
        description: errorMessage,
        variant: "destructive",
      });
    } finally {
      setIsLoading(false);
    }
  };

  // Handle password reset
  const handleResetPassword = async (e: React.FormEvent) => {
    e.preventDefault();

    if (!otp || !newPassword) {
      toast({
        title: "Missing Information",
        description: "Please enter both the verification code and new password",
        variant: "destructive",
      });
      return;
    }

    setIsLoading(true);

    try {
      // Verify the OTP
      await axios.post(`${API_URL}/users/verify-reset-code`, {
        email: verificationEmail || formData.email,
        code: otp
```

```
      });

      // Update the password
      await axios.post(`${API_URL}/users/reset-password`, {
        email: verificationEmail || formData.email,
        new_password: newPassword
      });

      toast({
        title: "Password Reset Successful",
        description: "Your password has been updated. You can now log in with your
new password.",
        variant: "default",
      });

      // Reset form state
      setIsForgotPassword(false);
      setOtp("");
      setNewPassword("");
      setFormData({ ...formData, password: "" });
    } catch (error) {
      const errorMessage =
        (error as any).response?.data?.detail ||
        (error as any).message ||
        "Failed to reset password. Please try again or request a new code.";

      toast({
        title: "Reset Failed",
        description: errorMessage,
        variant: "destructive",
      });
    } finally {
      setIsLoading(false);
    }
  };

  return (
    <div className="min-h-screen flex">
      <div className="hidden lg:flex lg:flex-1 relative">
        <Image
          src="https://media.istockphoto.com/id/1219569858/photo/woman-cooking-on-
the-modern-kitchen-at-home.jpg?
s=612x612&w=0&k=20&c=oUTIUAb0_cALWZ2GCTI3ZUmZCH31cf0UmBMPf1tMsck="
          alt="Smart Home"
          fill
          className="object-cover"
        />
        <div className="absolute inset-0 bg-black/20" />
        <div className="absolute bottom-20 left-10 text-white">
          <h1 className="text-5xl font-bold mb-4">
            <span className="text-[#00B2FF]">SYNC</span> your Home,
            <br />
```

```
            Save energy,
            <br />
            and live smarter.
          </h1>
          <Link href="/auth/signup">
            <Button className="bg-[#FF9500] hover:bg-[#FF9500]/90 text-lg px-8 py-
6">
              Sign Up
            </Button>
          </Link>
        </div>
      </div>

      <div className="flex-1 flex items-center justify-center p-8">
        <div className="w-full max-w-md">
          <div className="flex justify-end mb-8">
            <div className="flex items-center gap-2">
              <span className="text-2xl font-bold bg-[#00B2FF] text-white px-3 py-1
rounded-full">
                Sy<span className="text-[#FFB800]">nc</span>
              </span>
            </div>
          </div>

          <div className="mb-8">
            <h2 className="text-2xl font-semibold">Welcome,</h2>
            <p className="text-gray-600">Smart Home Login</p>
          </div>

          {!isForgotPassword ? (
            <form onSubmit={handleSubmit} className="space-y-6">
              <div className="space-y-2">
                <label className="text-sm font-medium">Email</label>
                <div className="relative">
                  <Input
                    type="email"
                    value={formData.email}
                    onChange={(e) => setFormData({ ...formData, email:
e.target.value })}
                    className="pl-10"
                    placeholder="Enter your email"
                    required
                  />
                  <Mail className="absolute left-3 top-1/2 -translate-y-1/2 h-5 w-5
text-gray-400" />
                </div>
              </div>

              <div className="space-y-2">
                <label className="text-sm font-medium">Password</label>
                <div className="relative">
                  <Input
```

```
                    type={showPassword ? "text" : "password"}
                    value={formData.password}
                    onChange={(e) => setFormData({ ...formData, password:
e.target.value })}
                    className="pr-10"
                    placeholder="Enter your password"
                    required
                  />
                  <button
                    type="button"
                    onClick={() => setShowPassword(!showPassword)}
                    className="absolute right-3 top-1/2 -translate-y-1/2"
                  >
                    {showPassword ? (
                      <EyeOff className="h-5 w-5 text-gray-400" />
                    ) : (
                      <Eye className="h-5 w-5 text-gray-400" />
                    )}
                  </button>
                </div>
              </div>

              <div className="flex justify-between items-center">
                <button
                  type="button"
                  onClick={handleForgotPassword}
                  className="text-sm text-[#00B2FF] hover:underline"
                >
                  Forgot Password?
                </button>
              </div>

              <Button
                type="submit"
                className="w-full bg-[#FF9500] hover:bg-[#FF9500]/90 py-6 text-lg"
                disabled={isLoading}
              >
                {isLoading ? 'Logging In...' : 'Login'}
              </Button>

              <div className="text-center space-y-2">
                <p className="text-sm text-gray-600">
                  {"Don't have an account? "}
                  <Link href="/auth/signup" className="text-[#00B2FF] bg-blue-50 px-
3 py-1 rounded-full text-sm hover:bg-blue-100">
                    Register Now
                  </Link>
                </p>
                <div className="flex items-center gap-2 justify-center">
                  <p className="text-sm text-gray-600">Part of a Household?</p>
                  <Link
                    href="/auth/household-login"
```

```
                  className="text-[#00B2FF] bg-blue-50 px-3 py-1 rounded-full
text-sm hover:bg-blue-100"
                >
                  Click Here
                </Link>
              </div>
            </div>
          </form>
        ) : (
          <form onSubmit={handleResetPassword} className="space-y-6">
            <div className="space-y-2">
              <label className="text-sm font-medium">Verification Code</label>
              <Input
                type="text"
                value={otp}
                onChange={(e) => setOtp(e.target.value)}
                placeholder="Enter the verification code from your email"
                required
              />
            </div>

            <div className="space-y-2">
              <label className="text-sm font-medium">New Password</label>
              <div className="relative">
                <Input
                  type={showPassword ? "text" : "password"}
                  value={newPassword}
                  onChange={(e) => setNewPassword(e.target.value)}
                  placeholder="Enter your new password"
                  required
                />
                <button
                  type="button"
                  onClick={() => setShowPassword(!showPassword)}
                  className="absolute right-3 top-1/2 -translate-y-1/2"
                >
                  {showPassword ? (
                    <EyeOff className="h-5 w-5 text-gray-400" />
                  ) : (
                    <Eye className="h-5 w-5 text-gray-400" />
                  )}
                </button>
              </div>
              <p className="text-xs text-gray-500">
                Password must be at least 8 characters with letters, numbers, and
special characters.
              </p>
            </div>

            <div className="flex space-x-2">
              <Button
                type="button"
```

```
                  className="flex-1 bg-gray-200 hover:bg-gray-300 text-gray-800"
                  onClick={() => setIsForgotPassword(false)}
                >
                  Back to Login
                </Button>
                <Button
                  type="submit"
                  className="flex-1 bg-[#FF9500] hover:bg-[#FF9500]/90"
                  disabled={isLoading}
                >
                  {isLoading ? 'Resetting...' : 'Reset Password'}
                </Button>
              </div>
            </form>
          )}
        </div>
      </div>
    </div>
  );
}
```

app/auth/login/page.tsx.bak.bak.bak

```
// login/page.tsx
"use client"

import Image from "next/image"
import { useState } from "react";
import { useRouter } from "next/navigation";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import Link from "next/link";
import { Eye, EyeOff, Mail } from "lucide-react";
import { toast } from "@/components/ui/use-toast";
import axios from 'axios';

// API base URL with fallback
const API_URL = process.env.NEXT_PUBLIC_API_URL || 'http://localhost:8000/api/v1';

export default function LoginPage() {
  const router = useRouter();
  const [showPassword, setShowPassword] = useState(false);
  const [formData, setFormData] = useState({ email: "", password: "" });
  const [isLoading, setIsLoading] = useState(false);
  const [isForgotPassword, setIsForgotPassword] = useState(false);
  const [otp, setOtp] = useState("");
  const [newPassword, setNewPassword] = useState("");
  const [verificationEmail, setVerificationEmail] = useState("");

  // Handle login form submission
```

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

  // Validate inputs
  if (!formData.email || !formData.password) {
    toast({
      title: "Missing Information",
      description: "Please enter both email and password",
      variant: "destructive",
    });
    return;
  }

  setIsLoading(true);

  try {
    // Using the correct API endpoint format from the documentation
    const response = await axios.post(`${API_URL}/users/token`,
      new URLSearchParams({
        'username': formData.email, // The API expects 'username' for email
        'password': formData.password
      }),
      {
        headers: {
          'Content-Type': 'application/x-www-form-urlencoded'
        }
      }
    );

    // Extract token and store it
    const { access_token, token_type } = response.data;

    // Store auth token
    localStorage.setItem('access_token', access_token);
    localStorage.setItem('token_type', token_type);

    // Store basic user info based on the token payload
    // Note: We're not fetching additional user data here
    localStorage.setItem('currentUser', JSON.stringify({
      email: formData.email,
      isAuthenticated: true,
      token: access_token
    }));

    toast({
      title: "Login Successful",
      description: "Welcome back!",
      variant: "default",
    });

    // Redirect to dashboard
    router.push("/dashboard");
```

```
  } catch (error) {
    // Handle error...
  } finally {
    setIsLoading(false);
  }
};

// Handle forgot password request
const handleForgotPassword = async () => {
  if (!formData.email) {
    toast({
      title: "Email Required",
      description: "Please enter your email address",
      variant: "destructive",
    });
    return;
  }

  setIsLoading(true);

  try {
    // Store the email for the verification step
    setVerificationEmail(formData.email);

    // Request password reset
    await axios.post(`${API_URL}/users/request-password-reset`, {
      email: formData.email
    });

    toast({
      title: "Verification Email Sent",
      description: "A confirmation link has been sent to your email",
      variant: "default",
    });

    setIsForgotPassword(true);
  } catch (error) {
    const errorMessage =
      (error as any).response?.data?.detail ||
      (error as any).message ||
      "Failed to process your request. The email may not be registered.";

    toast({
      title: "Request Failed",
      description: errorMessage,
      variant: "destructive",
    });
  } finally {
    setIsLoading(false);
  }
};
```

```
  // Handle password reset
  const handleResetPassword = async (e: React.FormEvent) => {
    e.preventDefault();

    if (!otp || !newPassword) {
      toast({
        title: "Missing Information",
        description: "Please enter both the verification code and new password",
        variant: "destructive",
      });
      return;
    }

    // Validate password
    const passwordRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[!@#$%^&*()_+\-=\[\]
{}|;':",.<>?/])[A-Za-z\d!@#$%^&*()_+\-=\[\]{}|;':",.<>?/]{8,}$/;
    if (!passwordRegex.test(newPassword)) {
      toast({
        title: "Invalid Password",
        description: "Password must be at least 8 characters with uppercase,
lowercase, numbers, and special characters.",
        variant: "destructive",
      });
      return;
    }

    setIsLoading(true);

    try {
      // Verify the OTP
      await axios.post(`${API_URL}/users/verify-reset-code`, {
        email: verificationEmail || formData.email,
        code: otp
      });

      // Update the password
      await axios.post(`${API_URL}/users/reset-password`, {
        email: verificationEmail || formData.email,
        new_password: newPassword
      });

      toast({
        title: "Password Reset Successful",
        description: "Your password has been updated. You can now log in with your
new password.",
        variant: "default",
      });

      // Reset form state
      setIsForgotPassword(false);
      setOtp("");
      setNewPassword("");
```

```
      setFormData({ ...formData, password: "" });
    } catch (error) {
      const errorMessage =
        (error as any).response?.data?.detail ||
        (error as any).message ||
        "Failed to reset password. Please try again or request a new code.";

      toast({
        title: "Reset Failed",
        description: errorMessage,
        variant: "destructive",
      });
    } finally {
      setIsLoading(false);
    }
  };

  return (
    <div className="min-h-screen flex">
      <div className="hidden lg:flex lg:flex-1 relative">
        <Image
          src="https://media.istockphoto.com/id/1219569858/photo/woman-cooking-on-
the-modern-kitchen-at-home.jpg?
s=612x612&w=0&k=20&c=oUTIUAb0_cALWZ2GCTI3ZUmZCH31cf0UmBMPf1tMsck="
          alt="Smart Home"
          fill
          className="object-cover"
        />
        <div className="absolute inset-0 bg-black/20" />
        <div className="absolute bottom-20 left-10 text-white">
          <h1 className="text-5xl font-bold mb-4">
            <span className="text-[#00B2FF]">SYNC</span> your Home,
            <br />
            Save energy,
            <br />
            and live smarter.
          </h1>
          <Link href="/auth/signup">
            <Button className="bg-[#FF9500] hover:bg-[#FF9500]/90 text-lg px-8 py-
6">
              Sign Up
            </Button>
          </Link>
        </div>
      </div>

      <div className="flex-1 flex items-center justify-center p-8">
        <div className="w-full max-w-md">
          <div className="flex justify-end mb-8">
            <div className="flex items-center gap-2">
              <span className="text-2xl font-bold bg-[#00B2FF] text-white px-3 py-1
rounded-full">
```

```
              Sy<span className="text-[#FFB800]">nc</span>
            </span>
          </div>
        </div>

        <div className="mb-8">
          <h2 className="text-2xl font-semibold">Welcome,</h2>
          <p className="text-gray-600">Smart Home Login</p>
        </div>

        {!isForgotPassword ? (
          <form onSubmit={handleSubmit} className="space-y-6">
            <div className="space-y-2">
              <label className="text-sm font-medium">Email</label>
              <div className="relative">
                <Input
                  type="email"
                  value={formData.email}
                  onChange={(e) => setFormData({ ...formData, email:
e.target.value })}
                  className="pl-10"
                  placeholder="Enter your email"
                  required
                />
                <Mail className="absolute left-3 top-1/2 -translate-y-1/2 h-5 w-5
text-gray-400" />
              </div>
            </div>

            <div className="space-y-2">
              <label className="text-sm font-medium">Password</label>
              <div className="relative">
                <Input
                  type={showPassword ? "text" : "password"}
                  value={formData.password}
                  onChange={(e) => setFormData({ ...formData, password:
e.target.value })}
                  className="pr-10"
                  placeholder="Enter your password"
                  required
                />
                <button
                  type="button"
                  onClick={() => setShowPassword(!showPassword)}
                  className="absolute right-3 top-1/2 -translate-y-1/2"
                >
                  {showPassword ? (
                    <EyeOff className="h-5 w-5 text-gray-400" />
                  ) : (
                    <Eye className="h-5 w-5 text-gray-400" />
                  )}
                </button>
```

```
              </div>
            </div>

            <div className="flex justify-between items-center">
              <button
                type="button"
                onClick={handleForgotPassword}
                className="text-sm text-[#00B2FF] hover:underline"
              >
                Forgot Password?
              </button>
            </div>

            <Button
              type="submit"
              className="w-full bg-[#FF9500] hover:bg-[#FF9500]/90 py-6 text-lg"
              disabled={isLoading}
            >
              {isLoading ? 'Logging In...' : 'Login'}
            </Button>

            <div className="text-center space-y-2">
              <p className="text-sm text-gray-600">
                {"Don't have an account? "}
                <Link href="/auth/signup" className="text-[#00B2FF] bg-blue-50 px-
3 py-1 rounded-full text-sm hover:bg-blue-100">
                  Register Now
                </Link>
              </p>
              <div className="flex items-center gap-2 justify-center">
                <p className="text-sm text-gray-600">Part of a Household?</p>
                <Link
                  href="/auth/household-login"
                  className="text-[#00B2FF] bg-blue-50 px-3 py-1 rounded-full
text-sm hover:bg-blue-100"
                >
                  Click Here
                </Link>
              </div>
            </div>
          </form>
        ) : (
          <form onSubmit={handleResetPassword} className="space-y-6">
            <div className="space-y-2">
              <label className="text-sm font-medium">Verification Code</label>
              <Input
                type="text"
                value={otp}
                onChange={(e) => setOtp(e.target.value)}
                placeholder="Enter the verification code"
                required
              />
```

```
              <p className="text-xs text-gray-500">
                Enter the code sent to your email
              </p>
            </div>

            <div className="space-y-2">
              <label className="text-sm font-medium">New Password</label>
              <div className="relative">
                <Input
                  type={showPassword ? "text" : "password"}
                  value={newPassword}
                  onChange={(e) => setNewPassword(e.target.value)}
                  placeholder="Enter your new password"
                  required
                />
                <button
                  type="button"
                  onClick={() => setShowPassword(!showPassword)}
                  className="absolute right-3 top-1/2 -translate-y-1/2"
                >
                  {showPassword ? (
                    <EyeOff className="h-5 w-5 text-gray-400" />
                  ) : (
                    <Eye className="h-5 w-5 text-gray-400" />
                  )}
                </button>
              </div>
              <p className="text-xs text-gray-500">
                Password must be at least 8 characters with letters, numbers, and
special characters.
              </p>
            </div>

            <div className="flex space-x-2">
              <Button
                type="button"
                className="flex-1 bg-gray-200 hover:bg-gray-300 text-gray-800"
                onClick={() => setIsForgotPassword(false)}
              >
                Back to Login
              </Button>
              <Button
                type="submit"
                className="flex-1 bg-[#FF9500] hover:bg-[#FF9500]/90"
                disabled={isLoading}
              >
                {isLoading ? 'Resetting...' : 'Reset Password'}
              </Button>
            </div>
          </form>
        )}
      </div>
```

```
      </div>
    </div>
  );
}
```

app/auth/signup/page.tsx

```tsx
// signup/page.tsx
"use client"
import Image from "next/image"
import { useState, useEffect } from "react"
import { useRouter } from "next/navigation"
import { Button } from "@/components/ui/button"
import { Input } from "@/components/ui/input"
import { Label } from "@/components/ui/label"
import Link from "next/link"
import { Eye, EyeOff, Mail, User, MapPin, ArrowLeft } from "lucide-react"
import { useToast } from "@/components/ui/use-toast"
import { RadioGroup, RadioGroupItem } from "@/components/ui/radio-group"
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from
"@/components/ui/select"
import axios from "axios"
import PhoneInput from "react-phone-input-2"
import "react-phone-input-2/lib/style.css"
import { getNames, getCode } from "country-list"

// Base API URL with fallback
const API_URL = process.env.NEXT_PUBLIC_API_URL || "http://localhost:8000/api/v1"

export default function SignupPage() {
  const { toast } = useToast()
  const router = useRouter()
  const [step, setStep] = useState(1)
  const [showPassword, setShowPassword] = useState(false)
  const [showConfirmPassword, setShowConfirmPassword] = useState(false)
  const [verificationCode, setVerificationCode] = useState(["", "", "", "", "", ""])
  const [countries, setCountries] = useState([])
  const [cities, setCities] = useState([])
  const [loadingCities, setLoadingCities] = useState(false)
  const [isLoading, setIsLoading] = useState(false)
  const [formData, setFormData] = useState({
    email: "",
    password: "",
    confirmPassword: "",
    firstName: "",
    lastName: "",
    phoneNumber: "",
    gender: "",
    birthDay: "",
    birthMonth: "",
```

```
    birthYear: "",
    country: "",
    city: "",
    address: "",
  })

  const dayOptions = Array.from({ length: 31 }, (_, i) => ({
    value: String(i + 1),
    label: i + 1
  }))

  const monthOptions = Array.from({ length: 12 }, (_, i) => ({
    value: String(i + 1),
    label: i + 1
  }))

  const currentYear = new Date().getFullYear()
  const yearOptions = Array.from({ length: 100 }, (_, i) => ({
    value: String(currentYear - i),
    label: currentYear - i
  }))

  useEffect(() => {
    const countryNames = getNames()
    const countryOptions = countryNames.map((name) => ({
      value: getCode(name),
      label: name
    }))
    setCountries(countryOptions.sort((a, b) => a.label.localeCompare(b.label)))
  }, [])

  useEffect(() => {
    const fetchCities = async () => {
      if (formData.country) {
        setLoadingCities(true)
        try {
          const response = await axios.get(
            `https://api.geonames.org/searchJSON?
country=${formData.country}&featureClass=P&maxRows=100&population=100000&username=demo`
          )
          const cityOptions = response.data.geonames
            .map((city) => ({
              value: city.name,
              label: city.name
            }))
            .sort((a, b) => a.label.localeCompare(b.label))
          setCities(cityOptions)
        } catch (error) {
          console.error("Error fetching cities:", error)
          toast({
            title: "Error",
            description: "Failed to fetch cities. Please try again.",
```

```
            variant: "destructive"
          })
          // Fallback to empty array if API fails
          setCities([])
        }
        setLoadingCities(false)
      }
    }

    if (formData.country) {
      fetchCities()
    }
  }, [formData.country, toast])

  const handleVerificationCodeChange = (index, value) => {
    // Only accept digits
    if (!/^\d*$/.test(value) && value !== "") return

    // Limit to one character per input
    if (value.length > 1) return

    const newCode = [...verificationCode]
    newCode[index] = value
    setVerificationCode(newCode)

    // Auto-focus next input when a digit is entered
    if (value && index < 5) {
      const nextInput = document.getElementById(`code-${index + 1}`)
      nextInput?.focus()
    }
  }

  const validatePassword = (password) => {
    // Match the backend password validation requirements
    if (password.length < 8) return "Password must be at least 8 characters long."
    if (!/\d/.test(password)) return "Password must contain at least 1 number."
    if (!/[a-z]/.test(password)) return "Password must contain at least 1 lowercase
letter."
    if (!/[A-Z]/.test(password)) return "Password must contain at least 1 uppercase
letter."
    if (!/[!@#$%^&*()_+\-=\[\]{}|;':",.<>?/]/.test(password)) return "Password must
contain at least 1 special character."
    return null
  }

  const handleRegister = async () => {
    // Validate email
    if (!formData.email || !/\S+@\S+\.\S+/.test(formData.email)) {
      toast({
        title: "Invalid Email",
        description: "Please enter a valid email address.",
        variant: "destructive"
```

```
      })
      return
    }

    // Validate password
    const passwordError = validatePassword(formData.password)
    if (passwordError) {
      toast({
        title: "Invalid Password",
        description: passwordError,
        variant: "destructive"
      })
      return
    }

    // Check if passwords match
    if (formData.password !== formData.confirmPassword) {
      toast({
        title: "Passwords Don't Match",
        description: "Please ensure that your passwords match.",
        variant: "destructive"
      })
      return
    }

    setIsLoading(true)

    try {
      // Register the new user using the FastAPI endpoint
      await axios.post(`${API_URL}/users/register`, {
        email: formData.email,
        password: formData.password,
        role: "user" // Default role is user
      })

      toast({
        title: "Registration Initiated",
        description: "A confirmation link has been sent to your email.",
      })

      setStep(2)
    } catch (error) {
      const errorMessage = error.response?.data?.detail || "Registration failed.
Please try again."
      toast({
        title: "Registration Failed",
        description: errorMessage,
        variant: "destructive"
      })
    } finally {
      setIsLoading(false)
    }
```

```
  }

  const handleVerifyOtp = async () => {
    const otp = verificationCode.join("")

    // Validate OTP format
    if (otp.length !== 6 || !/^\d+$/.test(otp)) {
      toast({
        title: "Invalid Code",
        description: "Please enter a valid 6-digit verification code.",
        variant: "destructive"
      })
      return
    }

    setIsLoading(true)

    try {
      // Verify email using the backend endpoint
      await axios.post(`${API_URL}/users/verify`, {
        email: formData.email,
        code: otp
      })

      toast({
        title: "Email Verified",
        description: "Your email has been successfully verified.",
      })

      setStep(3)
    } catch (error) {
      const errorMessage = error.response?.data?.detail || "Verification failed.
Please check the code and try again."
      toast({
        title: "Verification Failed",
        description: errorMessage,
        variant: "destructive"
      })
    } finally {
      setIsLoading(false)
    }
  }

  const handleResendOtp = async () => {
    setIsLoading(true)

    try {
      // Request a new verification email from the backend
      await axios.post(`${API_URL}/users/resend-verification`, {
        email: formData.email
      })
```

```
      toast({
        title: "Verification Email Resent",
        description: "A new confirmation link has been sent to your email.",
      })
    } catch (error) {
      const errorMessage = error.response?.data?.detail || "Failed to resend
verification email. Please try again."
      toast({
        title: "Request Failed",
        description: errorMessage,
        variant: "destructive"
      })
    } finally {
      setIsLoading(false)
    }
  }

  const validateBirthdate = () => {
    const dayNum = parseInt(formData.birthDay, 10)
    const monthNum = parseInt(formData.birthMonth, 10)
    const yearNum = parseInt(formData.birthYear, 10)
    const currentDate = new Date()
    const currentYear = currentDate.getFullYear()

    const errors = {}

    if (isNaN(dayNum) || dayNum < 1 || dayNum > 31) {
      errors.day = "Day must be between 1 and 31"
    }

    if (isNaN(monthNum) || monthNum < 1 || monthNum > 12) {
      errors.month = "Month must be between 1 and 12"
    }

    if (yearNum > currentYear) {
      errors.year = "Year cannot be in the future"
    } else if (yearNum < 1900) {
      errors.year = "Year must be after 1900"
    }

    // Calculate age
    const birthDate = new Date(yearNum, monthNum - 1, dayNum)
    const ageDate = new Date(Date.now() - birthDate.getTime())
    const age = Math.abs(ageDate.getUTCFullYear() - 1970)

    if (age < 18) {
      errors.age = "You must be at least 18 years old"
    }

    return errors
  }
```

```
const handleNextToStep5 = () => {
  // Validate required fields
  if (!formData.gender) {
    toast({
      title: "Gender Required",
      description: "Please select your gender.",
      variant: "destructive"
    })
    return
  }

  // Validate birthdate
  const errors = validateBirthdate()
  if (Object.keys(errors).length > 0) {
    Object.entries(errors).forEach(([key, message]) => {
      toast({
        title: `${key.charAt(0).toUpperCase() + key.slice(1)} Error`,
        description: message,
        variant: "destructive"
      })
    })
    return
  }

  setStep(5)
}

const handleCompleteRegistration = async () => {
  // Validate required location fields
  if (!formData.country || !formData.city || !formData.address.trim()) {
    toast({
      title: "Missing Information",
      description: "Please fill in all location fields.",
      variant: "destructive"
    })
    return
  }

  setIsLoading(true)

  try {
    // Create a complete profile by updating the user's profile
    const profileData = {
      user_id: formData.email, // Use email as identifier
      name: `${formData.firstName} ${formData.lastName}`,
      age: String(currentYear - parseInt(formData.birthYear)),
      profile_type: "adult", // Default profile type
      accessibility_settings: {
        preferred_language: "english",
        text_size: "normal",
      },
      can_control_devices: true,
```

```
        can_access_energy_data: true,
        can_manage_notifications: true,
        additional_info: {
          first_name: formData.firstName,
          last_name: formData.lastName,
          phone_number: formData.phoneNumber,
          gender: formData.gender,
          birthdate: {
            day: formData.birthDay,
            month: formData.birthMonth,
            year: formData.birthYear
          },
          address: {
            country: formData.country,
            city: formData.city,
            street_address: formData.address
          }
        }
      }

      // Create user profile
      await axios.post(`${API_URL}/profiles/create`, profileData)

      toast({
        title: "Registration Complete",
        description: "Your account has been created successfully!",
      })

      // Redirect to login page
      router.push("/auth/login")
    } catch (error) {
      const errorMessage = error.response?.data?.detail || "Failed to complete
registration. Please try again."
      toast({
        title: "Registration Failed",
        description: errorMessage,
        variant: "destructive"
      })
    } finally {
      setIsLoading(false)
    }
  }

  const renderStep = () => {
    switch (step) {
      case 1:
        return (
          <div className="space-y-6">
            <div className="space-y-2">
              <Label>Enter your Email</Label>
              <div className="relative">
                <Input
```

```
                  type="email"
                  value={formData.email}
                  onChange={(e) => setFormData({ ...formData, email: e.target.value
})}
                  className="pl-10"
                  placeholder="Enter your Email"
                  required
                />
                <Mail className="absolute left-3 top-1/2 -translate-y-1/2 h-5 w-5
text-gray-400" />
              </div>
            </div>

            <div className="space-y-2">
              <Label>Password</Label>
              <div className="relative">
                <Input
                  type={showPassword ? "text" : "password"}
                  value={formData.password}
                  onChange={(e) => setFormData({ ...formData, password:
e.target.value })}
                  className="pr-10"
                  placeholder="Enter your password"
                  required
                />
                <button
                  type="button"
                  onClick={() => setShowPassword(!showPassword)}
                  className="absolute right-3 top-1/2 -translate-y-1/2"
                >
                  {showPassword ? (
                    <EyeOff className="h-5 w-5 text-gray-400" />
                  ) : (
                    <Eye className="h-5 w-5 text-gray-400" />
                  )}
                </button>
              </div>
              <p className="text-xs text-gray-500">
                Password must be at least 8 characters with uppercase, lowercase,
numbers, and special characters.
              </p>
            </div>

            <div className="space-y-2">
              <Label>Confirm Password</Label>
              <div className="relative">
                <Input
                  type={showConfirmPassword ? "text" : "password"}
                  value={formData.confirmPassword}
                  onChange={(e) => setFormData({ ...formData, confirmPassword:
e.target.value })}
                  className="pr-10"
```

```
                placeholder="Confirm your password"
                required
              />
              <button
                type="button"
                onClick={() => setShowConfirmPassword(!showConfirmPassword)}
                className="absolute right-3 top-1/2 -translate-y-1/2"
              >
                {showConfirmPassword ? (
                  <EyeOff className="h-5 w-5 text-gray-400" />
                ) : (
                  <Eye className="h-5 w-5 text-gray-400" />
                )}
              </button>
            </div>
          </div>

          <Button
            className="w-full bg-[#FF9500] hover:bg-[#FF9500]/90 py-6 text-lg"
            onClick={handleRegister}
            disabled={isLoading}
          >
            {isLoading ? "Processing..." : "Next"}
          </Button>
        </div>
      )

    case 2:
      return (
        <div className="space-y-6">
          <h3 className="text-lg font-semibold">Verify Your Email</h3>
          <p className="text-sm text-gray-500">We've sent a confirmation link to
{formData.email}</p>

          <div className="bg-blue-50 p-4 rounded-md border border-blue-200">
            <p className="text-sm text-blue-800">
              Please check your inbox and click the verification link to confirm
your email address.
              This step is required to activate your account.
            </p>
          </div>

          <div className="bg-gray-50 p-4 rounded-md">
            <h4 className="text-sm font-medium mb-2">Can't find the email?</h4>
            <ul className="text-xs text-gray-600 space-y-1">
              <li>• Check your spam or junk folder</li>
              <li>• Make sure you entered the correct email address</li>
              <li>• Allow a few minutes for the email to arrive</li>
            </ul>
          </div>

          <p className="text-sm text-center">
```

```
              Didn't receive the email?{" "}
              <button
                className="text-[#00B2FF] hover:underline"
                onClick={handleResendOtp}
                disabled={isLoading}
              >
                Resend verification email
              </button>
            </p>

            <Button
              className="w-full bg-[#FF9500] hover:bg-[#FF9500]/90 py-6 text-lg"
              onClick={() => {
                toast({
                  title: "Please Verify Your Email",
                  description: "You need to click the link in your email before
continuing.",
                  variant: "destructive"
                });
              }}
            >
              I've Verified My Email
            </Button>
          </div>
        )

      case 3:
        return (
          <div className="space-y-6">
            <h3 className="text-lg font-semibold">Personal Details</h3>
            <p className="text-sm text-gray-500">Fill in your personal
information</p>

            <div className="space-y-4">
              <div className="space-y-2">
                <Label>First Name</Label>
                <div className="relative">
                  <Input
                    value={formData.firstName}
                    onChange={(e) => setFormData({ ...formData, firstName:
e.target.value })}
                    placeholder="Enter your First Name"
                    className="pl-10"
                    required
                  />
                  <User className="absolute left-3 top-1/2 -translate-y-1/2 h-5 w-5
text-gray-400" />
                </div>
              </div>

              <div className="space-y-2">
                <Label>Last Name</Label>
```

```
                <div className="relative">
                  <Input
                    value={formData.lastName}
                    onChange={(e) => setFormData({ ...formData, lastName:
e.target.value })}
                    placeholder="Enter your Last Name"
                    className="pl-10"
                    required
                  />
                  <User className="absolute left-3 top-1/2 -translate-y-1/2 h-5 w-5
text-gray-400" />
                </div>
              </div>

              <div className="space-y-2">
                <Label>Phone Number</Label>
                <PhoneInput
                  country={"us"}
                  value={formData.phoneNumber}
                  onChange={(phone) => setFormData({ ...formData, phoneNumber: phone
})}
                  inputClass="!w-full !py-2 !px-3 !text-base !h-10 !rounded-md
!border-input"
                  containerClass="!w-full"
                  buttonClass="!border-input !bg-background !rounded-l-md !h-10 !w-
12"
                  dropdownClass="!bg-background !text-foreground"
                  searchClass="!bg-background !text-foreground"
                />
              </div>
            </div>

            <Button
              className="w-full bg-[#FF9500] hover:bg-[#FF9500]/90 py-6 text-lg"
              onClick={() => setStep(4)}
              disabled={!formData.firstName || !formData.lastName ||
!formData.phoneNumber}
            >
              Next
            </Button>
          </div>
        )

      case 4:
        return (
          <div className="space-y-6">
            <h3 className="text-lg font-semibold">Additional Details</h3>
            <p className="text-sm text-gray-500">Please provide your gender and
birthdate</p>

            <div className="space-y-4">
              <div className="space-y-2">
```

```
                <Label>Gender</Label>
                <RadioGroup
                  value={formData.gender}
                  onValueChange={(value) => setFormData({ ...formData, gender: value
})}
                  className="flex gap-4"
                >
                  <div className="flex items-center space-x-2">
                    <RadioGroupItem value="male" id="male" />
                    <Label htmlFor="male">Male</Label>
                  </div>
                  <div className="flex items-center space-x-2">
                    <RadioGroupItem value="female" id="female" />
                    <Label htmlFor="female">Female</Label>
                  </div>
                  <div className="flex items-center space-x-2">
                    <RadioGroupItem value="other" id="other" />
                    <Label htmlFor="other">Other</Label>
                  </div>
                </RadioGroup>
              </div>

              <div className="space-y-2">
                <Label>Birthdate</Label>
                <div className="flex gap-4">
                  <Select
                    value={formData.birthDay}
                    onValueChange={(value) => setFormData({ ...formData, birthDay:
value })}
                  >
                    <SelectTrigger>
                      <SelectValue placeholder="Day" />
                    </SelectTrigger>
                    <SelectContent>
                      {dayOptions.map((option) => (
                        <SelectItem key={option.value} value={option.value}>
                          {option.label}
                        </SelectItem>
                      ))}
                    </SelectContent>
                  </Select>
                  <Select
                    value={formData.birthMonth}
                    onValueChange={(value) => setFormData({ ...formData, birthMonth:
value })}
                  >
                    <SelectTrigger>
                      <SelectValue placeholder="Month" />
                    </SelectTrigger>
                    <SelectContent>
                      {monthOptions.map((option) => (
                        <SelectItem key={option.value} value={option.value}>
```

```
                        {option.label}
                      </SelectItem>
                    ))}
                  </SelectContent>
                </Select>
                <Select
                  value={formData.birthYear}
                  onValueChange={(value) => setFormData({ ...formData, birthYear:
value })}
                >
                  <SelectTrigger>
                    <SelectValue placeholder="Year" />
                  </SelectTrigger>
                  <SelectContent>
                    {yearOptions.map((option) => (
                      <SelectItem key={option.value} value={option.value}>
                        {option.label}
                      </SelectItem>
                    ))}
                  </SelectContent>
                </Select>
              </div>
            </div>
          </div>

          <Button
            className="w-full bg-[#FF9500] hover:bg-[#FF9500]/90 py-6 text-lg"
            onClick={handleNextToStep5}
          >
            Next
          </Button>
        </div>
      )

    case 5:
      return (
        <div className="space-y-6">
          <h3 className="text-lg font-semibold">Location Details</h3>
          <p className="text-sm text-gray-500">Fill in your location
information</p>

          <div className="space-y-4">
            <div className="space-y-2">
              <Label>Country</Label>
              <Select
                value={formData.country}
                onValueChange={(value) => setFormData({ ...formData, country:
value, city: "" })}
              >
                <SelectTrigger>
                  <SelectValue placeholder="Choose your country" />
                </SelectTrigger>
```

```
                <SelectContent>
                  {countries.map((country) => (
                    <SelectItem key={country.value} value={country.value}>
                      {country.label}
                    </SelectItem>
                  ))}
                </SelectContent>
              </Select>
            </div>

            <div className="space-y-2">
              <Label>City</Label>
              {loadingCities ? (
                <div className="h-10 flex items-center justify-center bg-muted
rounded-md">
                  <p className="text-sm text-muted-foreground">Loading cities...
</p>
                </div>
              ) : (
                cities.length > 0 ? (
                  <Select
                    value={formData.city}
                    onValueChange={(value) => setFormData({ ...formData, city:
value })}
                    disabled={!formData.country}
                  >
                    <SelectTrigger>
                      <SelectValue placeholder="Choose your city" />
                    </SelectTrigger>
                    <SelectContent>
                      {cities.map((city) => (
                        <SelectItem key={city.value} value={city.value}>
                          {city.label}
                        </SelectItem>
                      ))}
                    </SelectContent>
                  </Select>
                ) : (
                  <Input
                    value={formData.city}
                    onChange={(e) => setFormData({ ...formData, city:
e.target.value })}
                    placeholder="Enter your city"
                    disabled={!formData.country}
                  />
                )
              )}
            </div>

            <div className="space-y-2">
              <Label>Address</Label>
              <div className="relative">
```

```
                  <Input
                    value={formData.address}
                    onChange={(e) => setFormData({ ...formData, address:
e.target.value })}
                    placeholder="Enter your address"
                    className="pl-10"
                  />
                  <MapPin className="absolute left-3 top-1/2 -translate-y-1/2 h-5 w-
5 text-gray-400" />
                </div>
              </div>
            </div>

            <Button
              className="w-full bg-[#FF9500] hover:bg-[#FF9500]/90 py-6 text-lg"
              onClick={handleCompleteRegistration}
              disabled={isLoading}
            >
              {isLoading ? "Completing Registration..." : "Complete Registration"}
            </Button>
          </div>
        )

      default:
        return null
    }
  }

  const handleBack = () => {
    if (step > 1) {
      setStep(step - 1)
    }
  }

  return (
    <div className="min-h-screen flex">
      <div className="hidden lg:flex lg:flex-1 relative">
        <Image
          src="https://media.istockphoto.com/id/1219569858/photo/woman-cooking-on-
the-modern-kitchen-at-home.jpg?
s=612x612&w=0&k=20&c=oUTIUAb0_cALWZ2GCTI3ZUmZCH31cf0UmBMPf1tMsck="
          alt="Smart Home"
          fill
          className="object-cover"
        />
        <div className="absolute inset-0 bg-black/20" />
        <div className="absolute bottom-20 left-10 text-white">
          <h1 className="text-5xl font-bold mb-4">
            <span className="text-[#00B2FF]">SYNC</span> your Home,
            <br />
            Save energy,
            <br />
```

```
            and live smarter.
          </h1>
        </div>
      </div>

      <div className="flex-1 flex items-center justify-center p-8">
        <div className="w-full max-w-md">
          <div className="flex justify-between items-center mb-8">
            {step > 1 && (
              <button
                onClick={handleBack}
                className="text-[#00B2FF] hover:underline flex items-center"
                disabled={isLoading}
              >
                <ArrowLeft className="mr-2" size={16} />
                Back
              </button>
            )}
            <div className="flex items-center gap-2 ml-auto">
              <span className="text-2xl font-bold bg-[#00B2FF] text-white px-3 py-1
rounded-full">
                Sy<span className="text-[#FFB800]">nc</span>
              </span>
            </div>
          </div>

          {renderStep()}

          {step === 1 && (
            <div className="mt-6 text-center">
              <p className="text-sm text-gray-600">
                Already have an account?{" "}
                <Link href="/auth/login" className="text-[#00B2FF] hover:underline">
                  Login Here
                </Link>
              </p>
            </div>
          )}
        </div>
      </div>
    </div>
  )
}
```

app/auth/signup/page.tsx.bak

```
// signup/page.tsx
"use client"
import Image from "next/image"
import { useState, useEffect } from "react"
```

```tsx
import { useRouter } from "next/navigation"
import { Button } from "@/components/ui/button"
import { Input } from "@/components/ui/input"
import { Label } from "@/components/ui/label"
import Link from "next/link"
import { Eye, EyeOff, Mail, User, MapPin, ArrowLeft } from "lucide-react"
import { useToast } from "@/components/ui/use-toast"
import { RadioGroup, RadioGroupItem } from "@/components/ui/radio-group"
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from
"@/components/ui/select"
import axios from "axios"
import PhoneInput from "react-phone-input-2"
import "react-phone-input-2/lib/style.css"
import { getNames, getCode } from "country-list"

// Base API URL with fallback
const API_URL = process.env.NEXT_PUBLIC_API_URL || "http://localhost:8000/api/v1"

export default function SignupPage() {
  const { toast } = useToast()
  const router = useRouter()
  const [step, setStep] = useState(1)
  const [showPassword, setShowPassword] = useState(false)
  const [showConfirmPassword, setShowConfirmPassword] = useState(false)
  const [verificationCode, setVerificationCode] = useState(["", "", "", "", "", ""])
  const [countries, setCountries] = useState([])
  const [cities, setCities] = useState([])
  const [loadingCities, setLoadingCities] = useState(false)
  const [isLoading, setIsLoading] = useState(false)
  const [formData, setFormData] = useState({
    email: "",
    password: "",
    confirmPassword: "",
    firstName: "",
    lastName: "",
    phoneNumber: "",
    gender: "",
    birthDay: "",
    birthMonth: "",
    birthYear: "",
    country: "",
    city: "",
    address: "",
  })

  const dayOptions = Array.from({ length: 31 }, (_, i) => ({
    value: String(i + 1),
    label: i + 1
  }))

  const monthOptions = Array.from({ length: 12 }, (_, i) => ({
    value: String(i + 1),
```

```
    label: i + 1
  }))

  const currentYear = new Date().getFullYear()
  const yearOptions = Array.from({ length: 100 }, (_, i) => ({
    value: String(currentYear - i),
    label: currentYear - i
  }))

  useEffect(() => {
    const countryNames = getNames()
    const countryOptions = countryNames.map((name) => ({
      value: getCode(name),
      label: name
    }))
    setCountries(countryOptions.sort((a, b) => a.label.localeCompare(b.label)))
  }, [])

  useEffect(() => {
    const fetchCities = async () => {
      if (formData.country) {
        setLoadingCities(true)
        try {
          const response = await axios.get(
            `https://api.geonames.org/searchJSON?
country=${formData.country}&featureClass=P&maxRows=100&population=100000&username=demo`
          )
          const cityOptions = response.data.geonames
            .map((city) => ({
              value: city.name,
              label: city.name
            }))
            .sort((a, b) => a.label.localeCompare(b.label))
          setCities(cityOptions)
        } catch (error) {
          console.error("Error fetching cities:", error)
          toast({
            title: "Error",
            description: "Failed to fetch cities. Please try again.",
            variant: "destructive"
          })
          // Fallback to empty array if API fails
          setCities([])
        }
        setLoadingCities(false)
      }
    }

    if (formData.country) {
      fetchCities()
    }
  }, [formData.country, toast])
```

```
  const handleVerificationCodeChange = (index, value) => {
    // Only accept digits
    if (!/^\d*$/.test(value) && value !== "") return

    // Limit to one character per input
    if (value.length > 1) return

    const newCode = [...verificationCode]
    newCode[index] = value
    setVerificationCode(newCode)

    // Auto-focus next input when a digit is entered
    if (value && index < 5) {
      const nextInput = document.getElementById(`code-${index + 1}`)
      nextInput?.focus()
    }
  }

  const validatePassword = (password) => {
    // Match the backend password validation requirements
    if (password.length < 8) return "Password must be at least 8 characters long."
    if (!/\d/.test(password)) return "Password must contain at least 1 number."
    if (!/[a-z]/.test(password)) return "Password must contain at least 1 lowercase
letter."
    if (!/[A-Z]/.test(password)) return "Password must contain at least 1 uppercase
letter."
    if (!/[!@#$%^&*()_+\-=\[\]{}|;':",.<>?/]/.test(password)) return "Password must
contain at least 1 special character."
    return null
  }

  const handleRegister = async () => {
    // Validate email
    if (!formData.email || !/\S+@\S+\.\S+/.test(formData.email)) {
      toast({
        title: "Invalid Email",
        description: "Please enter a valid email address.",
        variant: "destructive"
      })
      return
    }

    // Validate password
    const passwordError = validatePassword(formData.password)
    if (passwordError) {
      toast({
        title: "Invalid Password",
        description: passwordError,
        variant: "destructive"
      })
      return
```

```
    }

    // Check if passwords match
    if (formData.password !== formData.confirmPassword) {
      toast({
        title: "Passwords Don't Match",
        description: "Please ensure that your passwords match.",
        variant: "destructive"
      })
      return
    }

    setIsLoading(true)

    try {
      // Register the new user using the FastAPI endpoint
      await axios.post(`${API_URL}/users/register`, {
        email: formData.email,
        password: formData.password,
        role: "user" // Default role is user
      })

      toast({
        title: "Registration Initiated",
        description: "A verification code has been sent to your email.",
      })

      setStep(2)
    } catch (error) {
      const errorMessage = error.response?.data?.detail || "Registration failed.
Please try again."
      toast({
        title: "Registration Failed",
        description: errorMessage,
        variant: "destructive"
      })
    } finally {
      setIsLoading(false)
    }
  }

  const handleVerifyOtp = async () => {
    const otp = verificationCode.join("")

    // Validate OTP format
    if (otp.length !== 6 || !/^\d+$/.test(otp)) {
      toast({
        title: "Invalid Code",
        description: "Please enter a valid 6-digit verification code.",
        variant: "destructive"
      })
      return
```

```
    }

    setIsLoading(true)

    try {
      // Verify email using the backend endpoint
      await axios.post(`${API_URL}/users/verify`, {
        email: formData.email,
        code: otp
      })

      toast({
        title: "Email Verified",
        description: "Your email has been successfully verified.",
      })

      setStep(3)
    } catch (error) {
      const errorMessage = error.response?.data?.detail || "Verification failed.
Please check the code and try again."
      toast({
        title: "Verification Failed",
        description: errorMessage,
        variant: "destructive"
      })
    } finally {
      setIsLoading(false)
    }
  }

  const handleResendOtp = async () => {
    setIsLoading(true)

    try {
      // Request a new verification code from the backend
      await axios.post(`${API_URL}/users/resend-verification`, {
        email: formData.email
      })

      toast({
        title: "Verification Code Resent",
        description: "A new verification code has been sent to your email.",
      })

      // Reset the verification code inputs
      setVerificationCode(["", "", "", "", "", ""])
    } catch (error) {
      const errorMessage = error.response?.data?.detail || "Failed to resend
verification code. Please try again."
      toast({
        title: "Request Failed",
        description: errorMessage,
```

```
      variant: "destructive"
    })
  } finally {
    setIsLoading(false)
  }
}

const validateBirthdate = () => {
  const dayNum = parseInt(formData.birthDay, 10)
  const monthNum = parseInt(formData.birthMonth, 10)
  const yearNum = parseInt(formData.birthYear, 10)
  const currentDate = new Date()
  const currentYear = currentDate.getFullYear()

  const errors = {}

  if (isNaN(dayNum) || dayNum < 1 || dayNum > 31) {
    errors.day = "Day must be between 1 and 31"
  }

  if (isNaN(monthNum) || monthNum < 1 || monthNum > 12) {
    errors.month = "Month must be between 1 and 12"
  }

  if (yearNum > currentYear) {
    errors.year = "Year cannot be in the future"
  } else if (yearNum < 1900) {
    errors.year = "Year must be after 1900"
  }

  // Calculate age
  const birthDate = new Date(yearNum, monthNum - 1, dayNum)
  const ageDate = new Date(Date.now() - birthDate.getTime())
  const age = Math.abs(ageDate.getUTCFullYear() - 1970)

  if (age < 18) {
    errors.age = "You must be at least 18 years old"
  }

  return errors
}

const handleNextToStep5 = () => {
  // Validate required fields
  if (!formData.gender) {
    toast({
      title: "Gender Required",
      description: "Please select your gender.",
      variant: "destructive"
    })
    return
  }
```

```
  // Validate birthdate
  const errors = validateBirthdate()
  if (Object.keys(errors).length > 0) {
    Object.entries(errors).forEach(([key, message]) => {
      toast({
        title: `${key.charAt(0).toUpperCase() + key.slice(1)} Error`,
        description: message,
        variant: "destructive"
      })
    })
    return
  }

  setStep(5)
}

const handleCompleteRegistration = async () => {
  // Validate required location fields
  if (!formData.country || !formData.city || !formData.address.trim()) {
    toast({
      title: "Missing Information",
      description: "Please fill in all location fields.",
      variant: "destructive"
    })
    return
  }

  setIsLoading(true)

  try {
    // Create a complete profile by updating the user's profile
    const profileData = {
      user_id: formData.email, // Use email as identifier
      name: `${formData.firstName} ${formData.lastName}`,
      age: String(currentYear - parseInt(formData.birthYear)),
      profile_type: "adult", // Default profile type
      accessibility_settings: {
        preferred_language: "english",
        text_size: "normal",
      },
      can_control_devices: true,
      can_access_energy_data: true,
      can_manage_notifications: true,
      additional_info: {
        first_name: formData.firstName,
        last_name: formData.lastName,
        phone_number: formData.phoneNumber,
        gender: formData.gender,
        birthdate: {
          day: formData.birthDay,
          month: formData.birthMonth,
```

```
            year: formData.birthYear
          },
          address: {
            country: formData.country,
            city: formData.city,
            street_address: formData.address
          }
        }
      }

      // Create user profile
      await axios.post(`${API_URL}/profiles/create`, profileData)

      toast({
        title: "Registration Complete",
        description: "Your account has been created successfully!",
      })

      // Redirect to login page
      router.push("/auth/login")
    } catch (error) {
      const errorMessage = error.response?.data?.detail || "Failed to complete
registration. Please try again."
      toast({
        title: "Registration Failed",
        description: errorMessage,
        variant: "destructive"
      })
    } finally {
      setIsLoading(false)
    }
  }

  const renderStep = () => {
    switch (step) {
      case 1:
        return (
          <div className="space-y-6">
            <div className="space-y-2">
              <Label>Enter your Email</Label>
              <div className="relative">
                <Input
                  type="email"
                  value={formData.email}
                  onChange={(e) => setFormData({ ...formData, email: e.target.value
})}
                  className="pl-10"
                  placeholder="Enter your Email"
                  required
                />
                <Mail className="absolute left-3 top-1/2 -translate-y-1/2 h-5 w-5
text-gray-400" />
```

```
            </div>
          </div>

          <div className="space-y-2">
            <Label>Password</Label>
            <div className="relative">
              <Input
                type={showPassword ? "text" : "password"}
                value={formData.password}
                onChange={(e) => setFormData({ ...formData, password:
e.target.value })}
                  className="pr-10"
                  placeholder="Enter your password"
                  required
                />
                <button
                  type="button"
                  onClick={() => setShowPassword(!showPassword)}
                  className="absolute right-3 top-1/2 -translate-y-1/2"
                >
                  {showPassword ? (
                    <EyeOff className="h-5 w-5 text-gray-400" />
                  ) : (
                    <Eye className="h-5 w-5 text-gray-400" />
                  )}
                </button>
            </div>
            <p className="text-xs text-gray-500">
              Password must be at least 8 characters with uppercase, lowercase,
numbers, and special characters.
            </p>
          </div>

          <div className="space-y-2">
            <Label>Confirm Password</Label>
            <div className="relative">
              <Input
                type={showConfirmPassword ? "text" : "password"}
                value={formData.confirmPassword}
                onChange={(e) => setFormData({ ...formData, confirmPassword:
e.target.value })}
                  className="pr-10"
                  placeholder="Confirm your password"
                  required
                />
                <button
                  type="button"
                  onClick={() => setShowConfirmPassword(!showConfirmPassword)}
                  className="absolute right-3 top-1/2 -translate-y-1/2"
                >
                  {showConfirmPassword ? (
                    <EyeOff className="h-5 w-5 text-gray-400" />
```

```
                ) : (
                  <Eye className="h-5 w-5 text-gray-400" />
                )}
              </button>
            </div>
          </div>

          <Button
            className="w-full bg-[#FF9500] hover:bg-[#FF9500]/90 py-6 text-lg"
            onClick={handleRegister}
            disabled={isLoading}
          >
            {isLoading ? "Processing..." : "Next"}
          </Button>
        </div>
      )

    case 2:
      return (
        <div className="space-y-6">
          <h3 className="text-lg font-semibold">Verify Your Email</h3>
          <p className="text-sm text-gray-500">We've sent a 6-digit verification
code to {formData.email}</p>

          <div className="flex justify-between gap-2">
            {verificationCode.map((digit, index) => (
              <Input
                key={index}
                id={`code-${index}`}
                type="text"
                inputMode="numeric"
                maxLength={1}
                value={digit}
                onChange={(e) => handleVerificationCodeChange(index,
e.target.value)}
                className="w-10 h-12 text-center text-xl"
              />
            ))}
          </div>

          <p className="text-sm text-center">
            Didn't receive the code?{" "}
            <button
              className="text-[#00B2FF] hover:underline"
              onClick={handleResendOtp}
              disabled={isLoading}
            >
              Resend verification code
            </button>
          </p>

          <Button
```

```
              className="w-full bg-[#FF9500] hover:bg-[#FF9500]/90 py-6 text-lg"
              onClick={handleVerifyOtp}
              disabled={isLoading}
            >
              {isLoading ? "Verifying..." : "Verify & Continue"}
            </Button>
          </div>
        )

      case 3:
        return (
          <div className="space-y-6">
            <h3 className="text-lg font-semibold">Personal Details</h3>
            <p className="text-sm text-gray-500">Fill in your personal
information</p>

            <div className="space-y-4">
              <div className="space-y-2">
                <Label>First Name</Label>
                <div className="relative">
                  <Input
                    value={formData.firstName}
                    onChange={(e) => setFormData({ ...formData, firstName:
e.target.value })}
                    placeholder="Enter your First Name"
                    className="pl-10"
                    required
                  />
                  <User className="absolute left-3 top-1/2 -translate-y-1/2 h-5 w-5
text-gray-400" />
                </div>
              </div>

              <div className="space-y-2">
                <Label>Last Name</Label>
                <div className="relative">
                  <Input
                    value={formData.lastName}
                    onChange={(e) => setFormData({ ...formData, lastName:
e.target.value })}
                    placeholder="Enter your Last Name"
                    className="pl-10"
                    required
                  />
                  <User className="absolute left-3 top-1/2 -translate-y-1/2 h-5 w-5
text-gray-400" />
                </div>
              </div>

              <div className="space-y-2">
                <Label>Phone Number</Label>
                <PhoneInput
```

```
                  country={"us"}
                  value={formData.phoneNumber}
                  onChange={(phone) => setFormData({ ...formData, phoneNumber: phone
})}
                  inputClass="!w-full !py-2 !px-3 !text-base !h-10 !rounded-md
!border-input"
                  containerClass="!w-full"
                  buttonClass="!border-input !bg-background !rounded-l-md !h-10 !w-
12"
                  dropdownClass="!bg-background !text-foreground"
                  searchClass="!bg-background !text-foreground"
                />
              </div>
            </div>

            <Button
              className="w-full bg-[#FF9500] hover:bg-[#FF9500]/90 py-6 text-lg"
              onClick={() => setStep(4)}
              disabled={!formData.firstName || !formData.lastName ||
!formData.phoneNumber}
            >
              Next
            </Button>
          </div>
        )

      case 4:
        return (
          <div className="space-y-6">
            <h3 className="text-lg font-semibold">Additional Details</h3>
            <p className="text-sm text-gray-500">Please provide your gender and
birthdate</p>

            <div className="space-y-4">
              <div className="space-y-2">
                <Label>Gender</Label>
                <RadioGroup
                  value={formData.gender}
                  onValueChange={(value) => setFormData({ ...formData, gender: value
})}
                  className="flex gap-4"
                >
                  <div className="flex items-center space-x-2">
                    <RadioGroupItem value="male" id="male" />
                    <Label htmlFor="male">Male</Label>
                  </div>
                  <div className="flex items-center space-x-2">
                    <RadioGroupItem value="female" id="female" />
                    <Label htmlFor="female">Female</Label>
                  </div>
                  <div className="flex items-center space-x-2">
                    <RadioGroupItem value="other" id="other" />
```

```
                <Label htmlFor="other">Other</Label>
              </div>
            </RadioGroup>
          </div>

          <div className="space-y-2">
            <Label>Birthdate</Label>
            <div className="flex gap-4">
              <Select
                value={formData.birthDay}
                onValueChange={(value) => setFormData({ ...formData, birthDay:
value })}
              >
                <SelectTrigger>
                  <SelectValue placeholder="Day" />
                </SelectTrigger>
                <SelectContent>
                  {dayOptions.map((option) => (
                    <SelectItem key={option.value} value={option.value}>
                      {option.label}
                    </SelectItem>
                  ))}
                </SelectContent>
              </Select>
              <Select
                value={formData.birthMonth}
                onValueChange={(value) => setFormData({ ...formData, birthMonth:
value })}
              >
                <SelectTrigger>
                  <SelectValue placeholder="Month" />
                </SelectTrigger>
                <SelectContent>
                  {monthOptions.map((option) => (
                    <SelectItem key={option.value} value={option.value}>
                      {option.label}
                    </SelectItem>
                  ))}
                </SelectContent>
              </Select>
              <Select
                value={formData.birthYear}
                onValueChange={(value) => setFormData({ ...formData, birthYear:
value })}
              >
                <SelectTrigger>
                  <SelectValue placeholder="Year" />
                </SelectTrigger>
                <SelectContent>
                  {yearOptions.map((option) => (
                    <SelectItem key={option.value} value={option.value}>
                      {option.label}
```

```
                    </SelectItem>
                  ))}
                </SelectContent>
              </Select>
            </div>
          </div>
        </div>

        <Button
          className="w-full bg-[#FF9500] hover:bg-[#FF9500]/90 py-6 text-lg"
          onClick={handleNextToStep5}
        >
          Next
        </Button>
      </div>
    )

  case 5:
    return (
      <div className="space-y-6">
        <h3 className="text-lg font-semibold">Location Details</h3>
        <p className="text-sm text-gray-500">Fill in your location
information</p>

        <div className="space-y-4">
          <div className="space-y-2">
            <Label>Country</Label>
            <Select
              value={formData.country}
              onValueChange={(value) => setFormData({ ...formData, country:
value, city: "" })}
            >
              <SelectTrigger>
                <SelectValue placeholder="Choose your country" />
              </SelectTrigger>
              <SelectContent>
                {countries.map((country) => (
                  <SelectItem key={country.value} value={country.value}>
                    {country.label}
                  </SelectItem>
                ))}
              </SelectContent>
            </Select>
          </div>

          <div className="space-y-2">
            <Label>City</Label>
            {loadingCities ? (
              <div className="h-10 flex items-center justify-center bg-muted
rounded-md">
                <p className="text-sm text-muted-foreground">Loading cities...
</p>
```

```
                </div>
              ) : (
                cities.length > 0 ? (
                  <Select
                    value={formData.city}
                    onValueChange={(value) => setFormData({ ...formData, city:
value })}
                    disabled={!formData.country}
                  >
                    <SelectTrigger>
                      <SelectValue placeholder="Choose your city" />
                    </SelectTrigger>
                    <SelectContent>
                      {cities.map((city) => (
                        <SelectItem key={city.value} value={city.value}>
                          {city.label}
                        </SelectItem>
                      ))}
                    </SelectContent>
                  </Select>
                ) : (
                  <Input
                    value={formData.city}
                    onChange={(e) => setFormData({ ...formData, city:
e.target.value })}
                    placeholder="Enter your city"
                    disabled={!formData.country}
                  />
                )
              )}
            </div>

            <div className="space-y-2">
              <Label>Address</Label>
              <div className="relative">
                <Input
                  value={formData.address}
                  onChange={(e) => setFormData({ ...formData, address:
e.target.value })}
                  placeholder="Enter your address"
                  className="pl-10"
                />
                <MapPin className="absolute left-3 top-1/2 -translate-y-1/2 h-5 w-
5 text-gray-400" />
              </div>
            </div>
          </div>

          <Button
            className="w-full bg-[#FF9500] hover:bg-[#FF9500]/90 py-6 text-lg"
            onClick={handleCompleteRegistration}
            disabled={isLoading}
```

```
            >
              {isLoading ? "Completing Registration..." : "Complete Registration"}
            </Button>
          </div>
        )

      default:
        return null
    }
  }

  const handleBack = () => {
    if (step > 1) {
      setStep(step - 1)
    }
  }

  return (
    <div className="min-h-screen flex">
      <div className="hidden lg:flex lg:flex-1 relative">
        <Image
          src="https://media.istockphoto.com/id/1219569858/photo/woman-cooking-on-
the-modern-kitchen-at-home.jpg?
s=612x612&w=0&k=20&c=oUTIUAb0_cALWZ2GCTI3ZUmZCH31cf0UmBMPf1tMsck="
          alt="Smart Home"
          fill
          className="object-cover"
        />
        <div className="absolute inset-0 bg-black/20" />
        <div className="absolute bottom-20 left-10 text-white">
          <h1 className="text-5xl font-bold mb-4">
            <span className="text-[#00B2FF]">SYNC</span> your Home,
            <br />
            Save energy,
            <br />
            and live smarter.
          </h1>
        </div>
      </div>

      <div className="flex-1 flex items-center justify-center p-8">
        <div className="w-full max-w-md">
          <div className="flex justify-between items-center mb-8">
            {step > 1 && (
              <button
                onClick={handleBack}
                className="text-[#00B2FF] hover:underline flex items-center"
                disabled={isLoading}
              >
                <ArrowLeft className="mr-2" size={16} />
                Back
              </button>
```

```
            )}
            <div className="flex items-center gap-2 ml-auto">
              <span className="text-2xl font-bold bg-[#00B2FF] text-white px-3 py-1
rounded-full">
                Sy<span className="text-[#FFB800]">nc</span>
              </span>
            </div>
          </div>

          {renderStep()}

          {step === 1 && (
            <div className="mt-6 text-center">
              <p className="text-sm text-gray-600">
                Already have an account?{" "}
                <Link href="/auth/login" className="text-[#00B2FF] hover:underline">
                  Login Here
                </Link>
              </p>
            </div>
          )}
        </div>
      </div>
    </div>
  )
}
```

app/automations/page.tsx

```
// automations/page.tsx
"use client"

import React from "react"

import { useState, useEffect } from "react"
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"
import { Button } from "@/components/ui/button"
import { Switch } from "@/components/ui/switch"
import { Badge } from "@/components/ui/badge"
import { Clock, Sun, Moon, Home, Plus, Zap, Droplet, Thermometer, Trash2, Edit,
Repeat } from "lucide-react"
import { motion } from "framer-motion"
import {
  Dialog,
  DialogContent,
  DialogDescription,
  DialogFooter,
  DialogHeader,
  DialogTitle,
} from "@/components/ui/dialog"
```

```tsx
import { Input } from "@/components/ui/input"
import { Label } from "@/components/ui/label"
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from
"@/components/ui/select"
import { Checkbox } from "@/components/ui/checkbox"
import { toast } from "@/components/ui/use-toast"
import { NavigationSidebar } from "@/app/components/navigation-sidebar"; // Import
the navbar


interface Automation {
  id: number
  name: string
  description: string
  icon: string
  active: boolean
  schedule: {
    type: "once" | "daily" | "weekly" | "custom"
    startTime: string
    endTime: string
    daysOfWeek?: number[]
  }
  devices: string[]
}

interface Device {
  id: string
  name: string
  type: string
}

const automationIcons: Record<string, any> = {
  Sun,
  Moon,
  Home,
  Clock,
  Zap,
  Droplet,
  Thermometer,
}

const daysOfWeek = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"]

export default function AutomationsPage() {
  const [automations, setAutomations] = useState<Automation[]>([])
  const [devices, setDevices] = useState<Device[]>([])
  const [isDialogOpen, setIsDialogOpen] = useState(false)
  const [editingAutomation, setEditingAutomation] = useState<Automation | null>
(null)
  const [newAutomation, setNewAutomation] = useState<Automation>({
    id: 0,
    name: "",
```

```
    description: "",
    icon: "Clock",
    active: true,
    schedule: {
      type: "once",
      startTime: "",
      endTime: "",
    },
    devices: [],
  })

  useEffect(() => {
    const storedAutomations = JSON.parse(localStorage.getItem("automations") || "
[]")
    setAutomations(storedAutomations)
    const storedDevices = JSON.parse(localStorage.getItem("devices") || "[]")
    setDevices(storedDevices)
  }, [])

  const handleToggleAutomation = (id: number, active: boolean) => {
    const updatedAutomations = automations.map((automation) =>
      automation.id === id ? { ...automation, active } : automation,
    )
    setAutomations(updatedAutomations)
    localStorage.setItem("automations", JSON.stringify(updatedAutomations))
  }

  const handleSaveAutomation = () => {
    if (editingAutomation) {
      const updatedAutomations = automations.map((automation) =>
        automation.id === editingAutomation.id ? { ...newAutomation, id:
editingAutomation.id } : automation,
      )
      setAutomations(updatedAutomations)
      localStorage.setItem("automations", JSON.stringify(updatedAutomations))
      toast({
        title: "Automation Updated",
        description: "Your automation has been successfully updated.",
      })
    } else {
      const newAutomationItem = {
        ...newAutomation,
        id: Date.now(),
      }
      const updatedAutomations = [...automations, newAutomationItem]
      setAutomations(updatedAutomations)
      localStorage.setItem("automations", JSON.stringify(updatedAutomations))
      toast({
        title: "Automation Created",
        description: "Your new automation has been successfully added.",
      })
    }
```

```
    setIsDialogOpen(false)
    setEditingAutomation(null)
    setNewAutomation({
      id: 0,
      name: "",
      description: "",
      icon: "Clock",
      active: true,
      schedule: {
        type: "once",
        startTime: "",
        endTime: "",
      },
      devices: [],
    })
  }

  const handleDeleteAutomation = (id: number) => {
    if (window.confirm("Are you sure you want to delete this automation?")) {
      const updatedAutomations = automations.filter((automation) => automation.id
!== id)
      setAutomations(updatedAutomations)
      localStorage.setItem("automations", JSON.stringify(updatedAutomations))
      toast({
        title: "Automation Deleted",
        description: "The automation has been successfully removed.",
      })
    }
  }

  const handleEditAutomation = (automation: Automation) => {
    setEditingAutomation(automation)
    setNewAutomation(automation)
    setIsDialogOpen(true)
  }

  const formatSchedule = (schedule: Automation["schedule"]) => {
    if (!schedule || !schedule.type) {
      return "Schedule not set"
    }

    switch (schedule.type) {
      case "once":
        return `Once from ${schedule.startTime} to ${schedule.endTime}`
      case "daily":
        return `Daily from ${schedule.startTime} to ${schedule.endTime}`
      case "weekly":
        return `Weekly on ${schedule.daysOfWeek?.map((day) =>
daysOfWeek[day]).join(", ")} from ${schedule.startTime} to ${schedule.endTime}`
      case "custom":
        return `Custom schedule from ${schedule.startTime} to ${schedule.endTime}`
      default:
```

```
        return "Invalid schedule"
    }
  }

  const containerVariants = {
    hidden: { opacity: 0 },
    visible: {
      opacity: 1,
      transition: {
        staggerChildren: 0.1,
      },
    },
  }

  const itemVariants = {
    hidden: { y: 20, opacity: 0 },
    visible: {
      y: 0,
      opacity: 1,
    },
  }

  return (
    <div className="min-h-screen bg-gray-50 flex">
      <NavigationSidebar />
      <div className="flex-1 ml-[72px]">
        <motion.div initial="hidden" animate="visible" variants={containerVariants}
className="p-6">
          <motion.header variants={itemVariants} className="flex justify-between
items-center mb-6">
            <div className="flex items-center gap-2">
              <div className="w-12 h-12 bg-gradient-to-br from-[#FF9500] to-
[#FFB800] rounded-full flex items-center justify-center shadow-lg">
                <Zap className="h-6 w-6 text-white" />
              </div>
              <div>
                <h1 className="text-2xl font-bold text-gray-800">Automations</h1>
                <p className="text-sm text-gray-500">Manage your smart home
routines</p>
              </div>
            </div>
            {automations.length > 0 && (
              <Button className="bg-[#00B2FF] hover:bg-[#00B2FF]/90" onClick={() =>
setIsDialogOpen(true)}>
                <Plus className="w-4 h-4 mr-2" />
                New Automation
              </Button>
            )}
          </motion.header>

          <Dialog open={isDialogOpen} onOpenChange={setIsDialogOpen}>
            <DialogContent className="sm:max-w-[425px]">
```

```
              <DialogHeader>
                <DialogTitle>{editingAutomation ? "Edit Automation" : "Create New
Automation"}</DialogTitle>
                <DialogDescription>
                  {editingAutomation ? "Modify your automation settings." : "Set up
a new automation for your smart home."}
                </DialogDescription>
              </DialogHeader>
              <div className="grid gap-4 py-4">
                <div className="grid grid-cols-4 items-center gap-4">
                  <Label htmlFor="name" className="text-right">
                    Name
                  </Label>
                  <Input
                    id="name"
                    value={newAutomation.name}
                    onChange={(e) => setNewAutomation({ ...newAutomation, name:
e.target.value })}
                    className="col-span-3"
                  />
                </div>
                <div className="grid grid-cols-4 items-center gap-4">
                  <Label htmlFor="icon" className="text-right">
                    Icon
                  </Label>
                  <Select
                    value={newAutomation.icon}
                    onValueChange={(value) => setNewAutomation({ ...newAutomation,
icon: value })}
                  >
                    <SelectTrigger className="col-span-3">
                      <SelectValue placeholder="Select icon" />
                    </SelectTrigger>
                    <SelectContent>
                      {Object.keys(automationIcons).map((icon) => (
                        <SelectItem key={icon} value={icon}>
                          {icon}
                        </SelectItem>
                      ))}
                    </SelectContent>
                  </Select>
                </div>
                <div className="grid grid-cols-4 items-center gap-4">
                  <Label htmlFor="scheduleType" className="text-right">
                    Schedule Type
                  </Label>
                  <Select
                    value={newAutomation.schedule.type}
                    onValueChange={(value: "once" | "daily" | "weekly" | "custom")
=>
                      setNewAutomation({
                        ...newAutomation,
```

```
                      schedule: { ...newAutomation.schedule, type: value },
                    })
                  }
                >
                  <SelectTrigger className="col-span-3">
                    <SelectValue placeholder="Select schedule type" />
                  </SelectTrigger>
                  <SelectContent>
                    <SelectItem value="once">Once</SelectItem>
                    <SelectItem value="daily">Daily</SelectItem>
                    <SelectItem value="weekly">Weekly</SelectItem>
                    <SelectItem value="custom">Custom</SelectItem>
                  </SelectContent>
                </Select>
              </div>
              <div className="grid grid-cols-4 items-center gap-4">
                <Label htmlFor="startTime" className="text-right">
                  Start Time
                </Label>
                <Input
                  id="startTime"
                  type="time"
                  value={newAutomation.schedule.startTime}
                  onChange={(e) =>
                    setNewAutomation({
                      ...newAutomation,
                      schedule: { ...newAutomation.schedule, startTime:
e.target.value },
                    })
                  }
                  className="col-span-3"
                />
              </div>
              <div className="grid grid-cols-4 items-center gap-4">
                <Label htmlFor="endTime" className="text-right">
                  End Time
                </Label>
                <Input
                  id="endTime"
                  type="time"
                  value={newAutomation.schedule.endTime}
                  onChange={(e) =>
                    setNewAutomation({
                      ...newAutomation,
                      schedule: { ...newAutomation.schedule, endTime:
e.target.value },
                    })
                  }
                  className="col-span-3"
                />
              </div>
              {newAutomation.schedule.type === "weekly" && (
```

```
                  <div className="grid grid-cols-4 items-center gap-4">
                    <Label className="text-right">Days</Label>
                    <div className="col-span-3 flex flex-wrap gap-2">
                      {daysOfWeek.map((day, index) => (
                        <div key={day} className="flex items-center space-x-2">
                          <Checkbox
                            id={`day-${index}`}
                            checked=
{newAutomation.schedule.daysOfWeek?.includes(index)}
                            onCheckedChange={(checked) => {
                              const newDays = checked
                                ? [...(newAutomation.schedule.daysOfWeek || []),
index]
                                : newAutomation.schedule.daysOfWeek?.filter((d) => d
!== index);
                              setNewAutomation({
                                ...newAutomation,
                                schedule: { ...newAutomation.schedule, daysOfWeek:
newDays },
                              });
                            }}
                          />
                          <Label htmlFor={`day-${index}`}>{day}</Label>
                        </div>
                      ))}
                    </div>
                  </div>
                )}
                <div className="grid grid-cols-4 items-center gap-4">
                  <Label htmlFor="description" className="text-right">
                    Description
                  </Label>
                  <Input
                    id="description"
                    value={newAutomation.description}
                    onChange={(e) => setNewAutomation({ ...newAutomation,
description: e.target.value })}
                    className="col-span-3"
                  />
                </div>
                <div className="grid grid-cols-4 items-center gap-4">
                  <Label htmlFor="devices" className="text-right">
                    Devices
                  </Label>
                  <Select
                    value={newAutomation.devices[0] || ""}
                    onValueChange={(value) => setNewAutomation({ ...newAutomation,
devices: [value] })}
                  >
                    <SelectTrigger className="col-span-3">
                      <SelectValue placeholder="Select device" />
                    </SelectTrigger>
```

```
                    <SelectContent>
                      {devices.map((device) => (
                        <SelectItem key={device.id} value={device.id}>
                          {device.name}
                        </SelectItem>
                      ))}
                    </SelectContent>
                  </Select>
                </div>
              </div>
              <DialogFooter>
                <Button type="submit" onClick={handleSaveAutomation}>
                  {editingAutomation ? "Update automation" : "Save automation"}
                </Button>
              </DialogFooter>
            </DialogContent>
          </Dialog>

          <motion.div variants={containerVariants} className="grid grid-cols-1
md:grid-cols-2 lg:grid-cols-3 gap-6">
            {automations.length === 0 ? (
              <motion.div variants={itemVariants} className="col-span-full">
                <Card className="bg-white shadow-lg">
                  <CardContent className="p-6 flex flex-col items-center text-
center">
                    <div className="w-16 h-16 bg-blue-100 rounded-full flex items-
center justify-center mb-4">
                      <Zap className="w-8 h-8 text-blue-500" />
                    </div>
                    <h3 className="text-lg font-semibold mb-2">No Automations
Yet</h3>
                    <p className="text-gray-500 mb-4">
                      Create your first automation to start managing your smart home
routines.
                    </p>
                    <Button className="bg-[#00B2FF] hover:bg-[#00B2FF]/90" onClick=
{() => setIsDialogOpen(true)}>
                      <Plus className="w-4 h-4 mr-2" />
                      Add Your First Automation
                    </Button>
                  </CardContent>
                </Card>
              </motion.div>
            ) : (
              automations.map((automation) => (
                <motion.div key={automation.id} variants={itemVariants}>
                  <Card className={`overflow-hidden ${automation.active ? "" :
"opacity-50"}`}>
                    <CardHeader
                      className={`${automation.active ? "bg-gradient-to-r from-
[#00B2FF] to-[#0085FF]" : "bg-gray-400"} text-white`}
                    >
```

```
                      <CardTitle className="flex items-center justify-between">
                        <span>{automation.name}</span>
                        <Switch
                          checked={automation.active}
                          onCheckedChange={(checked) =>
handleToggleAutomation(automation.id, checked)}
                        />
                      </CardTitle>
                    </CardHeader>
                    <CardContent className="p-4">
                      <div className="flex items-center space-x-4 mb-4">
                        <div className={`p-2 rounded-full ${automation.active ? "bg-
blue-100" : "bg-gray-200"}`}>
                          {automation.icon &&
                            automationIcons[automation.icon] &&
                            React.createElement(automationIcons[automation.icon], {
                              className: `w-6 h-6 ${automation.active ? "text-blue-
600" : "text-gray-600"}`,
                            })}
                        </div>
                        <div>
                          <p className="text-sm text-gray-600">
{automation.description}</p>
                          <div className="flex items-center space-x-2 mt-1">
                            <Repeat className="w-4 h-4 text-gray-400" />
                            <span className="text-xs text-gray-500">
{formatSchedule(automation.schedule)}</span>
                          </div>
                        </div>
                      </div>
                      <div className="flex items-center justify-between">
                        <Badge variant="outline" className="text-blue-600 border-
blue-600">
                          {automation.devices.length} devices
                        </Badge>
                        <div className="flex space-x-2">
                          <Button variant="outline" size="sm" onClick={() =>
handleEditAutomation(automation)}>
                            <Edit className="w-4 h-4 text-blue-500" />
                          </Button>
                          <Button variant="outline" size="sm" onClick={() =>
handleDeleteAutomation(automation.id)}>
                            <Trash2 className="w-4 h-4 text-red-500" />
                          </Button>
                        </div>
                      </div>
                    </CardContent>
                  </Card>
                </motion.div>
              ))
            )}
          </motion.div>
```

```
        </motion.div>
      </div>
    </div>
  );
}
```

app/components/active-devices.tsx

```
"use client"

import { useState, useEffect } from "react"
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"
import { Button } from "@/components/ui/button"
import { Switch } from "@/components/ui/switch"
import { Lightbulb, Thermometer, Fan, Tv, Maximize2, Minimize2 } from "lucide-react"
import styles from './ActiveDevices.module.css'

interface Device {
  id: string
  name: string
  type: string
  room: string
  status: "on" | "off"
}

const deviceIcons: Record<string, any> = {
  light: Lightbulb,
  thermostat: Thermometer,
  fan: Fan,
  tv: Tv,
}

export function ActiveDevices() {
  const [devices, setDevices] = useState<Device[]>([])
  const [isMaximized, setIsMaximized] = useState(false)

  useEffect(() => {
    const storedDevices = JSON.parse(localStorage.getItem("devices") || "[]")
    setDevices(storedDevices)
  }, [])

  const toggleDevice = (id: string) => {
    setDevices(
      devices.map((device) => {
        if (device.id === id) {
          const updatedDevice = {
            ...device,
            status: device.status === "on" ? "off" : "on",
          }
          // Update in localStorage
```

```
          const allDevices = JSON.parse(localStorage.getItem("devices") || "[]")
          const updatedDevices = allDevices.map((d: Device) => (d.id === id ?
updatedDevice : d))
          localStorage.setItem("devices", JSON.stringify(updatedDevices))
          return updatedDevice
        }
        return device
      }),
    )
  }

  return (
    <Card className={`${isMaximized ? styles.cardMaximizedA : ""} h-full`}>
      <CardHeader className={styles.cardTitleA}>
        <CardTitle>Active Devices</CardTitle>
        <Button variant="ghost" size="sm" onClick={() =>
setIsMaximized(!isMaximized)}>
          {isMaximized ? <Minimize2 className="h-4 w-4" /> : <Maximize2
className="h-4 w-4" />}
        </Button>
      </CardHeader>
      <CardContent>
        <div className={styles.deviceListA}>
          {devices.length === 0 ? (
            <p className="text-sm text-muted-foreground">No devices added yet</p>
          ) : (
            devices.slice(0, isMaximized ? devices.length : 4).map((device) => {
              const Icon = deviceIcons[device.type] || Lightbulb
              return (
                <div key={device.id} className={styles.deviceItemA}>
                  <div className="flex items-center space-x-4">
                    <div
                      className={`${styles.deviceIconWrapperA} ${device.status ===
"on" ? styles.deviceIconOn : styles.deviceIconOff}`}
                    >
                      <Icon className={`${styles.deviceIconA} ${device.status ===
"on" ? "text-[#00B2FF]" : "text-gray-500"}`} />
                    </div>
                    <div>
                      <p className={styles.deviceNameA}>{device.name}</p>
                      <p className={styles.deviceRoomA}>{device.room}</p>
                    </div>
                  </div>
                  <Switch
                    checked={device.status === "on"}
                    onCheckedChange={() => toggleDevice(device.id)}
                    className={styles.deviceStatusSwitchA}
                  />
                </div>
              )
            })
          )}
```

```
        </div>
      </CardContent>
    </Card>
  )
}
```

app/components/air-conditioner-control.tsx

```
"use client"

import { Card } from "@/components/ui/card"
import styles from './AirConditionerControl.module.css'

export function AirConditionerControl() {
  return (
    <Card className={styles.cardAir}>
      <h3 className={styles.cardTitleAir}>Air Conditioner</h3>
      {/* Add air conditioner control UI here */}
    </Card>
  )
}
```

app/components/automation-widget.tsx

```
"use client"

import { useState, useEffect } from "react"
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"
import { Button } from "@/components/ui/button"
import { Switch } from "@/components/ui/switch"
import { Badge } from "@/components/ui/badge"
import { Sun, Moon, Home, Clock, Plus } from "lucide-react"
import {
  Dialog,
  DialogContent,
  DialogDescription,
  DialogFooter,
  DialogHeader,
  DialogTitle,
  DialogTrigger,
} from "@/components/ui/dialog"
import { Input } from "@/components/ui/input"
import { Label } from "@/components/ui/label"
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from
"@/components/ui/select"
import styles from './AutomationWidget.module.css'

interface Automation {
```

```
  id: number
  name: string
  description: string
  icon: string
  active: boolean
  schedule: string
  type: string
}

const automationIcons: Record<string, any> = {
  Sun,
  Moon,
  Home,
  Clock,
}

export function AutomationWidget() {
  const [automations, setAutomations] = useState<Automation[]>([])
  const [newAutomation, setNewAutomation] = useState({
    name: "",
    description: "",
    icon: "Clock",
    schedule: "",
    type: "",
  })

  useEffect(() => {
    const storedAutomations = JSON.parse(localStorage.getItem("automations") || "
[]")
    setAutomations(storedAutomations)
  }, [])

  const toggleAutomation = (id: number) => {
    const updatedAutomations = automations.map((automation) =>
      automation.id === id ? { ...automation, active: !automation.active } :
automation,
    )
    setAutomations(updatedAutomations)
    localStorage.setItem("automations", JSON.stringify(updatedAutomations))
  }

  const addAutomation = () => {
    const newAutomationItem = {
      id: Date.now(),
      ...newAutomation,
      active: true,
    }
    const updatedAutomations = [...automations, newAutomationItem]
    setAutomations(updatedAutomations)
    localStorage.setItem("automations", JSON.stringify(updatedAutomations))
    setNewAutomation({
      name: "",
```

```
      description: "",
      icon: "Clock",
      schedule: "",
      type: "",
    })
  }

  return (
    <Card className={styles.cardAuto}>
      <CardHeader className={styles.cardHeaderAuto}>
        <CardTitle className={styles.cardTitleAuto}>Automations</CardTitle>
        <Dialog>
          <DialogTrigger asChild>
            <Button variant="outline" size="icon">
              <Plus className="h-4 w-4" />
            </Button>
          </DialogTrigger>
          <DialogContent className={styles.dialogContentAuto}>
            <DialogHeader>
              <DialogTitle>Create New Automation</DialogTitle>
              <DialogDescription>Set up a new automation for your smart home.
</DialogDescription>
            </DialogHeader>
            <div className={styles.gridContainerAuto}>
              <div className={styles.gridItemAuto}>
                <Label htmlFor="name" className="text-left">
                  Name
                </Label>
                <Input
                  id="name"
                  value={newAutomation.name}
                  onChange={(e) => setNewAutomation({ ...newAutomation, name:
e.target.value })}
                  className="col-span-3"
                />
              </div>
              <div className={styles.gridItemAuto}>
                <Label htmlFor="description" className="text-left">
                  Description
                </Label>
                <Input
                  id="description"
                  value={newAutomation.description}
                  onChange={(e) => setNewAutomation({ ...newAutomation, description:
e.target.value })}
                  className="col-span-3"
                />
              </div>
              <div className={styles.gridItemAuto}>
                <Label htmlFor="icon" className="text-left">
                  Icon
                </Label>
```

```
                <Select
                  value={newAutomation.icon}
                  onValueChange={(value) => setNewAutomation({ ...newAutomation,
icon: value })}
                >
                  <SelectTrigger className="col-span-3">
                    <SelectValue placeholder="Select icon" />
                  </SelectTrigger>
                  <SelectContent>
                    <SelectItem value="Sun">Sun</SelectItem>
                    <SelectItem value="Moon">Moon</SelectItem>
                    <SelectItem value="Home">Home</SelectItem>
                    <SelectItem value="Clock">Clock</SelectItem>
                  </SelectContent>
                </Select>
              </div>
              <div className={styles.gridItemAuto}>
                <Label htmlFor="schedule" className="text-left">
                  Schedule
                </Label>
                <Input
                  id="schedule"
                  value={newAutomation.schedule}
                  onChange={(e) => setNewAutomation({ ...newAutomation, schedule:
e.target.value })}
                  className="col-span-3"
                  placeholder="e.g., Daily at 8:00 AM"
                />
              </div>
              <div className={styles.gridItemAuto}>
                <Label htmlFor="type" className="text-left">
                  Type
                </Label>
                <Select
                  value={newAutomation.type}
                  onValueChange={(value) => setNewAutomation({ ...newAutomation,
type: value })}
                >
                  <SelectTrigger className="col-span-3">
                    <SelectValue placeholder="Select type" />
                  </SelectTrigger>
                  <SelectContent>
                    <SelectItem value="Time-based">Time-based</SelectItem>
                    <SelectItem value="Event-based">Event-based</SelectItem>
                    <SelectItem value="Condition-based">Condition-based</SelectItem>
                  </SelectContent>
                </Select>
              </div>
            </div>
            <DialogFooter>
              <Button type="submit" onClick={addAutomation}>
                Add Automation
```

```
              </Button>
            </DialogFooter>
          </DialogContent>
        </Dialog>
      </CardHeader>
      <CardContent className={styles.cardContentAuto}>
        <div className="space-y-4">
          {automations.length === 0 ? (
            <p className="text-sm text-muted-foreground">No automations set up
yet</p>
          ) : (
            automations.map((automation) => {
              const Icon = automationIcons[automation.icon] || Clock
              return (
                <div key={automation.id} className={styles.automationItemAuto}>
                  <div className={styles.detailsWrapperAuto}>
                    <div className={`${styles.iconWrapperAuto} ${automation.active ?
styles.activeIconAuto : styles.inactiveIconAuto}`}>
                      <Icon className={styles.iconAuto} />
                    </div>
                    <div>
                      <p className={styles.detailsTextAuto}>{automation.name}</p>
                      <p className={styles.scheduleTextAuto}>{automation.schedule}
</p>
                    </div>
                  </div>
                  <div className={styles.badgeWrapperAuto}>
                    <Badge variant="outline">{automation.type}</Badge>
                    <Switch checked={automation.active} onCheckedChange={() =>
toggleAutomation(automation.id)} />
                  </div>
                </div>
              )
            })
          )}
        </div>
      </CardContent>
    </Card>
  )
}
```

app/components/charts.tsx

```
"use client"

import { Line, Bar, Pie } from "react-chartjs-2"
import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
```

```
  PointElement,
  LineElement,
  BarElement,
  ArcElement,
  Title,
  Tooltip,
  Legend,
} from "chart.js"
import './charts.css'; // Import the CSS file

ChartJS.register(CategoryScale, LinearScale, PointElement, LineElement, BarElement,
ArcElement, Title, Tooltip, Legend)

export function LineChart({ className }: { className?: string }) {
  const data = {
    labels: ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"],
    datasets: [
      {
        data: [12, 19, 15, 17, 22, 28, 25],
        borderColor: "#00B2FF",
        backgroundColor: "rgba(0, 178, 255, 0.1)",
        tension: 0.4,
        fill: true,
      },
    ],
  }

  return (
    <div className={`${className} chart-containerC chart-lineC`}>
      <Line
        data={data}
        options={{
          responsive: true,
          plugins: {
            legend: {
              display: false,
            },
          },
        }}
      />
    </div>
  )
}

export function BarChart({ className }: { className?: string }) {
  const data = {
    labels: ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"],
    datasets: [
      {
        data: [15, 30, 25, 18, 12, 25, 28],
        backgroundColor: "#00B2FF",
      },
```

```
    ],
  }

  return (
    <div className={`${className} chart-containerC chart-barC`}>
      <Bar
        data={data}
        options={{
          responsive: true,
          plugins: {
            legend: {
              display: false,
            },
          },
        }}
      />
    </div>
  )
}

export function PieChart({ className }: { className?: string }) {
  const data = {
    labels: ["Heating", "Cooling", "Lighting", "Appliances", "Other"],
    datasets: [
      {
        data: [30, 25, 20, 15, 10],
        backgroundColor: ["#FF6384", "#36A2EB", "#FFCE56", "#4BC0C0", "#9966FF"],
      },
    ],
  }

  return (
    <div className={`${className} chart-containerC chart-pieC`}>
      <Pie
        data={data}
        options={{
          responsive: true,
          plugins: {
            legend: {
              position: "bottom" as const,
            },
          },
        }}
      />
    </div>
  )
}
```

app/components/device-control.tsx

```tsx
"use client"

import { Card, CardHeader, CardTitle, CardContent } from "@/components/ui/card"
import { Switch } from "@/components/ui/switch"
import { Slider } from "@/components/ui/slider"
import { Label } from "@/components/ui/label"
import { Lightbulb, Thermometer, Wind } from "lucide-react"
import './DeviceControl.module.css'; // Import the CSS file

interface DeviceControlProps {
  room: string
}

export function DeviceControl({ room }: DeviceControlProps) {
  return (
    <Card className="card-containerD">
      <CardHeader className="card-headerD">
        <CardTitle className="card-titleD">{room} Devices</CardTitle>
      </CardHeader>
      <CardContent className="card-contentD space-y-4">
        <div className="device-control-itemD">
          <div className="flex items-center space-x-2">
            <Lightbulb className="iconD" />
            <Label htmlFor="lights" className="labelD">Lights</Label>
          </div>
          <Switch id="lights" />
        </div>

        <div className="space-y-2">
          <div className="temperature-slider-containerD">
            <div className="flex items-center space-x-2">
              <Thermometer className="iconD" />
              <Label htmlFor="temperature" className="labelD">Temperature</Label>
            </div>
            <span>22°C</span>
          </div>
          <Slider id="temperature" min={16} max={30} step={1} defaultValue={[22]}
className="sliderD" />
        </div>

        <div className="device-control-itemD">
          <div className="flex items-center space-x-2">
            <Wind className="iconD" />
            <Label htmlFor="ac" className="labelD">Air Conditioning</Label>
          </div>
          <Switch id="ac" />
        </div>
      </CardContent>
    </Card>
```

```
  )
}
```

app/components/energy-consumption.tsx

```
"use client"

import { useState, useEffect } from "react"
import { Card, CardHeader, CardTitle, CardContent } from "@/components/ui/card"
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from
"@/components/ui/select"
import { LineChart } from "@/components/charts"
import './EnergyConsumption.module.css' // Import the CSS file

interface EnergyData {
  date: string
  usage: number
}

export function EnergyConsumption() {
  const [timeframe, setTimeframe] = useState("week")
  const [energyData, setEnergyData] = useState<EnergyData[]>([])

  useEffect(() => {
    // In a real application, this would fetch data from an API
    const generateData = () => {
      const data: EnergyData[] = []
      const now = new Date()
      const daysToGenerate = timeframe === "week" ? 7 : 30

      for (let i = daysToGenerate - 1; i >= 0; i--) {
        const date = new Date(now.getTime() - i * 24 * 60 * 60 * 1000)
        data.push({
          date: date.toISOString().split("T")[0],
          usage: Math.floor(Math.random() * 20) + 10, // Random usage between 10 and
30 kWh
        })
      }

      setEnergyData(data)
    }

    generateData()
  }, [timeframe])

  return (
    <Card className="card-container-E">
      <CardHeader className="card-header-E">
        <CardTitle className="card-title-E">Energy Consumption</CardTitle>
        <Select value={timeframe} onValueChange={(value) => setTimeframe(value)}>
```

```
          <SelectTrigger className="select-trigger-E">
            <SelectValue placeholder="Select timeframe" />
          </SelectTrigger>
          <SelectContent>
            <SelectItem value="week">Last Week</SelectItem>
            <SelectItem value="month">Last Month</SelectItem>
          </SelectContent>
        </Select>
      </CardHeader>
      <CardContent className="card-content-E">
        <LineChart
          data={energyData.map((d) => ({ name: d.date, usage: d.usage }))}
          xAxisLabel="Date"
          yAxisLabel="Energy (kWh)"
        />
      </CardContent>
    </Card>
  )
}
```

app/components/family-members.tsx

```
"use client"

import { useEffect, useState } from "react"
import { Card, CardHeader, CardTitle, CardContent } from "@/components/ui/card"
import { Avatar, AvatarFallback } from "@/components/ui/avatar"
import { Button } from "@/components/ui/button"
import Link from "next/link"
import './FamilyMembers.css' // Import the CSS file

interface FamilyMember {
  id: string
  name: string
  email: string
  role: string
}

export function FamilyMembers() {
  const [familyMembers, setFamilyMembers] = useState<FamilyMember[]>([])

  useEffect(() => {
    const storedMembers = JSON.parse(localStorage.getItem("familyMembers") || "[]")
    setFamilyMembers(storedMembers)
  }, [])

  return (
    <Card className="card-containerF">
      <CardHeader className="card-headerF">
        <CardTitle className="card-titleF">Family Members</CardTitle>
```

```
        <Link href="/profile">
          <Button variant="ghost" className="h-8 w-8 p-0">
            <span className="sr-only">Manage family members</span>
            <svg
              xmlns="http://www.w3.org/2000/svg"
              viewBox="0 0 24 24"
              fill="none"
              stroke="currentColor"
              strokeLinecap="round"
              strokeLinejoin="round"
              strokeWidth="2"
              className="button-iconF"
            >
              <path d="M12 5v14M5 12h14" />
            </svg>
          </Button>
        </Link>
      </CardHeader>
      <CardContent className="card-contentF">
        <div className="space-y-4">
          {familyMembers.length === 0 ? (
            <p className="text-sm text-muted-foreground">No family members added
yet.</p>
          ) : (
            familyMembers.map((member) => (
              <div key={member.id} className="family-memberF">
                <Avatar>
                  <AvatarFallback>{member.name[0]}</AvatarFallback>
                </Avatar>
                <div>
                  <p className="family-member-nameF">{member.name}</p>
                  <p className="family-member-roleF">{member.role}</p>
                </div>
              </div>
            ))
          )}
        </div>
      </CardContent>
    </Card>
  )
}
```

app/components/logout.ts

```
"use client"

import { useRouter } from "next/navigation"

export function useLogout() {
  const router = useRouter()
```

```
  return () => {
    // Clear localStorage items
    localStorage.removeItem('currentUser');
    localStorage.removeItem('access_token');
    localStorage.removeItem('token_type');

    // Use the correct path
    router.push("/auth/login")
  }
}
```

app/components/navigation-sidebar.tsx

```
"use client"

import { useState } from "react"
import { Home, Lightbulb, Clock, BarChart2, Plus, User, Settings, Smartphone } from
"lucide-react"
import { cn } from "@/lib/utils"
import { usePathname, useRouter } from "next/navigation"
import Link from "next/link"
import { AccessDeniedDialog } from "@/components/access-denied-dialog"

const navigationItems = [
  { icon: Home, path: "/", label: "Dashboard" },
  { icon: Lightbulb, path: "/suggestions", label: "Suggestions" },
  { icon: Clock, path: "/automations", label: "Automations", requiredPermission:
"addAutomation" },
  { icon: BarChart2, path: "/statistics", label: "Statistics", requiredPermission:
"statisticalData" },
  { icon: Smartphone, path: "/devices", label: "Devices" },
  { icon: Plus, path: "/add-room", label: "Add Room" },
  { icon: User, path: "/profile", label: "Profile" },
  { icon: Settings, path: "/settings", label: "Settings" },
]

export function NavigationSidebar() {
  const pathname = usePathname()
  const router = useRouter()
  const [accessDeniedOpen, setAccessDeniedOpen] = useState(false)
  const [deniedFeature, setDeniedFeature] = useState("")

  const handleNavigation = (path: string, requiredPermission?: string) => {
    // TODO: Add user role restricted access
    router.push(path)
  }

  return (
    <>
```

```
      <div className="fixed left-0 top-0 h-screen w-[72px] bg-[#FF9500] flex flex-
col items-center py-4 space-y-6">
        <Link href="/">
          <div className="w-[52px] h-[32px] bg-gradient-to-r from-[#00B2FF] to-
[#0085FF] rounded-full flex items-center justify-center">
            <span className="text-white font-bold text-sm">
              Sy<span className="text-[#FFB800]">nc</span>
            </span>
          </div>
        </Link>
        {navigationItems.map((item) => (
          <button
            key={item.path}
            onClick={() => handleNavigation(item.path, item.requiredPermission)}
            className={cn(
              "w-10 h-10 rounded-xl flex items-center justify-center",
              pathname === item.path ? "bg-white" : "hover:bg-white/10",
            )}
          >
            <item.icon className={cn("w-5 h-5", pathname === item.path ? "text-
[#FF9500]" : "text-white")} />
          </button>
        ))}
      </div>
      <AccessDeniedDialog
        isOpen={accessDeniedOpen}
        onClose={() => setAccessDeniedOpen(false)}
        featureName={deniedFeature}
      />
    </>
  )
}
```

app/components/room-energy-consumption.tsx

```
"use client"

import { useState } from "react"
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"
import { Battery, ChevronLeft, ChevronRight } from "lucide-react"
import { Button } from "@/components/ui/button"

interface Room {
  id: string
  name: string
  consumption: number
}

interface RoomEnergyConsumptionProps {
```

```
  rooms: Room[]
}

export function RoomEnergyConsumption({ rooms }: RoomEnergyConsumptionProps) {
  const [currentRoomIndex, setCurrentRoomIndex] = useState(0)

  const currentRoom = rooms[currentRoomIndex]

  const handlePrevRoom = () => {
    setCurrentRoomIndex((prevIndex) => (prevIndex > 0 ? prevIndex - 1 : rooms.length
- 1))
  }

  const handleNextRoom = () => {
    setCurrentRoomIndex((prevIndex) => (prevIndex < rooms.length - 1 ? prevIndex + 1
: 0))
  }

  const getConsumptionColor = (consumption: number) => {
    if (consumption < 20) return "bg-green-500"
    if (consumption < 50) return "bg-yellow-500"
    return "bg-red-500"
  }

  return (
    <Card className="col-span-full">
      <CardHeader className="flex flex-row items-center justify-between">
        <CardTitle>Room Energy Consumption</CardTitle>
        <div className="flex items-center space-x-2">
          <Button variant="outline" size="icon" onClick={handlePrevRoom}>
            <ChevronLeft className="h-4 w-4" />
          </Button>
          <Button variant="outline" size="icon" onClick={handleNextRoom}>
            <ChevronRight className="h-4 w-4" />
          </Button>
        </div>
      </CardHeader>
      <CardContent>
        <div className="flex items-center justify-between">
          <div className="flex items-center space-x-4">
            <Battery className="h-8 w-8 text-[#00B2FF]" />
            <div>
              <h3 className="text-lg font-semibold">{currentRoom.name}</h3>
              <p className="text-sm text-gray-500">Current consumption</p>
            </div>
          </div>
          <div className="text-right">
            <p className="text-2xl font-bold">{currentRoom.consumption} kWh</p>
            <p className="text-sm text-gray-500">Today's usage</p>
          </div>
        </div>
        <div className="mt-4">
```

```
          <div className="h-4 w-full bg-gray-200 rounded-full overflow-hidden">
            <div
              className={`h-full ${getConsumptionColor(currentRoom.consumption)}`}
              style={{ width: `${Math.min((currentRoom.consumption / 100) * 100,
100)}%` }}
            ></div>
          </div>
        </div>
        <div className="mt-2 flex justify-between text-sm text-gray-500">
          <span>0 kWh</span>
          <span>50 kWh</span>
          <span>100 kWh</span>
        </div>
      </CardContent>
    </Card>
  )
}
```

app/components/room-preview.tsx

```
import { Card, CardHeader, CardTitle, CardContent } from "@/components/ui/card"
import Image from "next/image"

interface RoomPreviewProps {
  room: string
}

export function RoomPreview({ room }: RoomPreviewProps) {
  return (
    <Card className="mt-4">
      <CardHeader>
        <CardTitle>{room} Preview</CardTitle>
      </CardHeader>
      <CardContent>
        <div className="relative h-48 rounded-md overflow-hidden">
          <Image
            src="https://hebbkx1anhila5yf.public.blob.vercel-storage.com/IMG_0697-
MRkZkq9qJMNVm20BYSDpYOdkYHt3pF.jpeg"
            alt={`${room} Preview`}
            layout="fill"
            objectFit="cover"
          />
        </div>
      </CardContent>
    </Card>
  )
}
```

app/components/room-selector.tsx

```
"use client"

import { useEffect, useState } from "react"
import { Button } from "@/components/ui/button"
import { ChevronRight } from "lucide-react"

interface RoomSelectorProps {
  selectedRoom: string
  onSelectRoom: (room: string) => void
}

export function RoomSelector({ selectedRoom, onSelectRoom }: RoomSelectorProps) {
  const [rooms, setRooms] = useState(["Living Room", "Bedroom", "Kitchen",
"Bathroom", "Office", "Kids Room"])

  useEffect(() => {
    const storedRooms = JSON.parse(localStorage.getItem("rooms") || "[]")
    if (storedRooms.length > 0) {
      setRooms((prevRooms) => [...prevRooms, ...storedRooms.map((room: any) =>
room.name)])
    }
  }, [])

  return (
    <div className="flex items-center space-x-2 overflow-x-auto py-2">
      {rooms.map((room) => (
        <Button
          key={room}
          variant={room === selectedRoom ? "default" : "ghost"}
          className={room === selectedRoom ? "bg-[#00B2FF] hover:bg-[#00B2FF]/90" :
""}
          onClick={() => onSelectRoom(room)}
        >
          {room}
        </Button>
      ))}
      <Button variant="ghost" size="icon">
        <ChevronRight className="h-4 w-4" />
      </Button>
    </div>
  )
}
```

app/components/smart-scenes.tsx

```tsx
import { Card, CardHeader, CardTitle, CardContent } from "@/components/ui/card"
import { Button } from "@/components/ui/button"
import { Sun, Moon, Coffee, Tv, Book, DoorClosed } from "lucide-react"

const scenes = [
  { name: "Good Morning", icon: Sun, color: "bg-yellow-500" },
  { name: "Good Night", icon: Moon, color: "bg-blue-500" },
  { name: "Movie Time", icon: Tv, color: "bg-purple-500" },
  { name: "Reading Mode", icon: Book, color: "bg-green-500" },
  { name: "Coffee Break", icon: Coffee, color: "bg-orange-500" },
  { name: "Away Mode", icon: DoorClosed, color: "bg-red-500" },
]

export function SmartScenes() {
  return (
    <Card>
      <CardHeader>
        <CardTitle>Smart Scenes</CardTitle>
      </CardHeader>
      <CardContent>
        <div className="grid grid-cols-2 gap-3">
          {scenes.map((scene) => (
            <Button key={scene.name} variant="outline" className="h-auto py-4 flex
flex-col items-center gap-2">
              <div className={`p-2 rounded-full ${scene.color}`}>
                <scene.icon className="h-5 w-5 text-white" />
              </div>
              <span className="text-sm">{scene.name}</span>
            </Button>
          ))}
        </div>
      </CardContent>
    </Card>
  )
}
```

app/components/sortable-widget.tsx

```tsx
"use client"

import { useSortable } from "@dnd-kit/sortable"
import { CSS } from "@dnd-kit/utilities"
import { GripVertical } from "lucide-react"
import type React from "react"

interface SortableWidgetProps {
  id: string
  isEditing: boolean
```

```
  children: React.ReactNode
}

export function SortableWidget({ id, isEditing, children }: SortableWidgetProps) {
  const { attributes, listeners, setNodeRef, transform, transition, isDragging } =
useSortable({ id })

  const style = {
    transform: CSS.Transform.toString(transform),
    transition,
    zIndex: isDragging ? 1000 : "auto",
    opacity: isDragging ? 0.5 : 1,
  }

  return (
    <div ref={setNodeRef} style={style} className="relative group">
      {isEditing && (
        <div
          className="absolute right-2 top-2 cursor-move z-50 opacity-0 group-
hover:opacity-100 transition-opacity"
          {...attributes}
          {...listeners}
        >
          <GripVertical className="h-5 w-5 text-gray-400" />
        </div>
      )}
      {children}
    </div>
  )
}
```

app/components/time-date-widget.tsx

```
"use client"

import { useState, useEffect } from "react"
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"
import { Clock } from "lucide-react"

export function TimeDateWidget() {
  const [currentTime, setCurrentTime] = useState(new Date())

  useEffect(() => {
    const timer = setInterval(() => setCurrentTime(new Date()), 1000)
    return () => clearInterval(timer)
  }, [])

  const formatDate = (date: Date) => {
    return date.toLocaleDateString("en-US", {
```

```
      weekday: "long",
      year: "numeric",
      month: "long",
      day: "numeric",
    })
  }

  const formatTime = (date: Date) => {
    return date.toLocaleTimeString("en-US", {
      hour: "2-digit",
      minute: "2-digit",
      second: "2-digit",
    })
  }

  return (
    <Card>
      <CardHeader className="flex flex-row items-center justify-between space-y-0
pb-2">
        <CardTitle className="text-sm font-medium">Current Time & Date</CardTitle>
        <Clock className="h-4 w-4 text-muted-foreground" />
      </CardHeader>
      <CardContent>
        <div className="text-2xl font-bold">{formatTime(currentTime)}</div>
        <p className="text-xs text-muted-foreground">{formatDate(currentTime)}</p>
      </CardContent>
    </Card>
  )
}
```

app/components/weather-widget.tsx

```
import { Card, CardHeader, CardTitle, CardContent } from "@/components/ui/card"
import { Sun, Cloud, Droplets } from "lucide-react"

export function WeatherWidget() {
  return (
    <Card>
      <CardHeader>
        <CardTitle>Weather</CardTitle>
      </CardHeader>
      <CardContent>
        <div className="flex items-center justify-between">
          <div className="flex items-center space-x-2">
            <Sun className="h-8 w-8 text-yellow-500" />
            <div>
              <p className="text-2xl font-bold">28°C</p>
              <p className="text-sm text-gray-500">Sunny</p>
            </div>
```

```
          </div>
          <div>
            <div className="flex items-center space-x-1">
              <Cloud className="h-4 w-4" />
              <span className="text-sm">20%</span>
            </div>
            <div className="flex items-center space-x-1">
              <Droplets className="h-4 w-4" />
              <span className="text-sm">30%</span>
            </div>
          </div>
        </div>
      </CardContent>
    </Card>
  )
}
```

app/create-profile/page.tsx

```
// create-profile/page.tsx
"use client";

import { useState } from "react";
import Image from "next/image";
import { useRouter } from "next/navigation";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Label } from "@/components/ui/label";
import {
    Select,
    SelectContent,
    SelectItem,
    SelectTrigger,
    SelectValue,
} from "@/components/ui/select";
import { Switch } from "@/components/ui/switch";
import { ArrowLeft, Upload, Users } from "lucide-react";
import { RadioGroup, RadioGroupItem } from "@/components/ui/radio-group";
import { motion } from "framer-motion";
import { toast } from "@/components/ui/use-toast";
import axios from "axios";

const API_URL = process.env.NEXT_PUBLIC_API_URL || "http://localhost:8000";

export default function CreateProfilePage() {
    const router = useRouter();
    const [step, setStep] = useState(1);
    const [formData, setFormData] = useState({
```

```
        profileType: "",
        role: "",
        firstName: "",
        lastName: "",
        email: "",
        age: "",
        energyGoal: "",
        accessibility: false,
        avatar: "",
        pinType: "admin", // "admin" or "user"
        pin: ["", "", "", ""],
        permissions: {
            notifications: true,
            energyAlerts: true,
            addAutomation: false,
            statisticalData: false,
            deviceControl: true,
            roomControl: false,
        },
    });

    const handlePinChange = (index: number, value: string) => {
        if (value.length > 1) return; // Only allow single digits
        const newPin = [...formData.pin];
        newPin[index] = value;
        setFormData({ ...formData, pin: newPin });

        // Auto-focus next input
        if (value !== "" && index < 3) {
            const nextInput = document.getElementById(`pin-${index + 1}`);
            nextInput?.focus();
        }
    };

    const handleSubmit = async () => {
        const pin = formData.pin.join("");
        const currentUser = JSON.parse(localStorage.getItem("currentUser") || "{}");

        const payload = {
            name: `${formData.firstName} ${formData.lastName}`,
            email: formData.email,
            account_type: formData.role,
            pin_option: formData.pinType,
            household_id: currentUser.householdId, // use householdId from current
user
            admin_user: currentUser.email, // use admin user from current user
            pin: pin,
            permissions: formData.permissions,
        };
        try {
            await axios.post(`${API_URL}/api/create-profile`, payload);
```

```
            toast({
                title: "Profile Created",
                description:
                    "The new household member profile has been successfully
created.",
            });
            router.push("/manage-profiles");
        } catch (error) {
            console.error("Error creating profile:", error);
            toast({
                title: "Error",
                description: "Failed to create profile. Please try again.",
                variant: "destructive",
            });
        }
    };

    return (
        <div className="min-h-screen bg-gradient-to-br from-gray-50 to-gray-100 p-
6">
            <motion.div
                initial={{ opacity: 0, y: -20 }}
                animate={{ opacity: 1, y: 0 }}
                transition={{ duration: 0.5 }}
            >
                <header className="flex justify-between items-center mb-8">
                    <div className="flex items-center gap-4">
                        <Button variant="ghost" size="icon" onClick={() =>
router.back()}>
                            <ArrowLeft className="h-6 w-6" />
                        </Button>
                        <div className="flex items-center gap-4">
                            <div className="w-12 h-12 bg-gradient-to-br from-
[#00B2FF] to-[#0085FF] rounded-full flex items-center justify-center shadow-lg">
                                <Users className="h-6 w-6 text-white" />
                            </div>
                            <div>
                                <h1 className="text-2xl font-bold text-gray-800">
                                    Create Profile
                                </h1>
                                <p className="text-sm text-gray-500">
                                    Add a new family member or guest
                                </p>
                            </div>
                        </div>
                    </div>
                </header>
            </motion.div>

            <motion.div
                initial={{ opacity: 0, y: 20 }}
                animate={{ opacity: 1, y: 0 }}
```

```
                transition={{ duration: 0.5 }}
            >
                <Card className="max-w-2xl mx-auto">
                    <CardHeader>
                        <CardTitle>
                            {step === 1
                                ? "Profile Information"
                                : step === 2
                                    ? "Set PIN Code"
                                    : "Set Permissions"}
                        </CardTitle>
                    </CardHeader>
                    <CardContent>
                        {step === 1 && (
                            <div className="grid grid-cols-2 gap-6">
                                <div className="col-span-2">
                                    <Label>Select Profile Type</Label>
                                    <Select
                                        value={formData.profileType}
                                        onValueChange={(value) =>
                                            setFormData({ ...formData, profileType:
value })
                                        }
                                    >
                                        <SelectTrigger>
                                            <SelectValue placeholder="Select profile
type" />
                                        </SelectTrigger>
                                        <SelectContent>
                                            <SelectItem value="family">Family
Member</SelectItem>
                                            <SelectItem
value="guest">Guest</SelectItem>
                                        </SelectContent>
                                    </Select>
                                </div>

                                <div className="col-span-2">
                                    <Label>Select Role</Label>
                                    <Select
                                        value={formData.role}
                                        onValueChange={(value) =>
                                            setFormData({ ...formData, role: value
})
                                        }
                                    >
                                        <SelectTrigger>
                                            <SelectValue placeholder="Select role"
/>
                                        </SelectTrigger>
                                        <SelectContent>
                                            <SelectItem
```

```
value="adult">Adult</SelectItem>
                                            <SelectItem
value="child">Child</SelectItem>
                                            <SelectItem
value="admin">Admin</SelectItem>
                                        </SelectContent>
                                    </Select>
                                </div>

                                <div>
                                    <Label>First Name</Label>
                                    <Input
                                        value={formData.firstName}
                                        onChange={(e) =>
                                            setFormData({ ...formData, firstName:
e.target.value })
                                        }
                                    />
                                </div>

                                <div>
                                    <Label>Last Name</Label>
                                    <Input
                                        value={formData.lastName}
                                        onChange={(e) =>
                                            setFormData({ ...formData, lastName:
e.target.value })
                                        }
                                    />
                                </div>

                                <div className="col-span-2">
                                    <Label>Email Address</Label>
                                    <Input
                                        type="email"
                                        value={formData.email}
                                        onChange={(e) =>
                                            setFormData({ ...formData, email:
e.target.value })
                                        }
                                    />
                                </div>

                                <div>
                                    <Label>Age</Label>
                                    <Input
                                        type="number"
                                        value={formData.age}
                                        onChange={(e) =>
                                            setFormData({ ...formData, age:
e.target.value })
                                        }
```

```
                                    />
                                </div>

                                <div>
                                    <Label>Energy Saving Goal (kWh)</Label>
                                    <Input
                                        type="number"
                                        value={formData.energyGoal}
                                        onChange={(e) =>
                                            setFormData({ ...formData, energyGoal:
e.target.value })
                                        }
                                    />
                                </div>

                                <div className="col-span-2">
                                    <div className="flex items-center justify-
between">
                                        <Label>Accessibility features</Label>
                                        <Switch
                                            checked={formData.accessibility}
                                            onCheckedChange={(checked) =>
                                                setFormData({ ...formData,
accessibility: checked })
                                            }
                                        />
                                    </div>
                                </div>

                                <div className="col-span-2">
                                    <Label>Profile Avatar</Label>
                                    <div className="mt-2 flex flex-col items-center
justify-center border-2 border-dashed rounded-xl p-6">
                                        <div className="w-24 h-24 bg-gray-200
rounded-full flex items-center justify-center mb-4">
                                            {formData.avatar ? (
                                                <Image
                                                    src={formData.avatar ||
"/placeholder.svg"}
                                                    alt="Avatar"
                                                    width={96}
                                                    height={96}
                                                    className="rounded-full"
                                                />
                                            ) : (
                                                <Upload className="w-8 h-8 text-
gray-400" />
                                            )}
                                        </div>
                                        <Button
                                            variant="outline"
                                            onClick={() =>
```

```
                                        document.getElementById("avatar-
upload")?.click()
                                    }
                                >
                                    Select Avatar
                                </Button>
                                <input
                                    id="avatar-upload"
                                    type="file"
                                    className="hidden"
                                    accept="image/*"
                                    onChange={(e) => {
                                        const file = e.target.files?.[0];
                                        if (file) {
                                            const reader = new FileReader();
                                            reader.onloadend = () => {
                                                setFormData({
                                                    ...formData,
                                                    avatar: reader.result as
string,
                                                });
                                            };
                                            reader.readAsDataURL(file);
                                        }
                                    }}
                                />
                            </div>
                        </div>

                        <div className="col-span-2">
                            <Button
                                className="w-full bg-[#00B2FF] hover:bg-
[#00B2FF]/90"
                                onClick={() => setStep(2)}
                            >
                                Next
                            </Button>
                        </div>
                    </div>
                )}

                {step === 2 && (
                    <div className="space-y-6">
                        <div className="space-y-4">
                            <Label>PIN Setup Method</Label>
                            <RadioGroup
                                defaultValue={formData.pinType}
                                onValueChange={(value) =>
                                    setFormData({ ...formData, pinType:
value })
                                }
                                className="flex flex-col space-y-2"
```

```
                                    >
                                        <div className="flex items-center space-x-
2">
                                            <RadioGroupItem value="admin" id="admin"
/>
                                            <Label htmlFor="admin">I set the
PIN</Label>
                                        </div>
                                        <div className="flex items-center space-x-
2">
                                            <RadioGroupItem value="user" id="user"
/>
                                            <Label htmlFor="user">User sets the
PIN</Label>
                                        </div>
                                    </RadioGroup>
                                </div>

                                {formData.pinType === "admin" && (
                                    <div className="space-y-4">
                                        <Label>Enter 4-digit PIN</Label>
                                        <div className="flex justify-center gap-4">
                                            {[0, 1, 2, 3].map((i) => (
                                                <Input
                                                    key={i}
                                                    id={`pin-${i}`}
                                                    type="text"
                                                    inputMode="numeric"
                                                    pattern="[0-9]*"
                                                    maxLength={1}
                                                    className="w-16 h-16 text-2xl
text-center"
                                                    value={formData.pin[i]}
                                                    onChange={(e) =>
                                                        handlePinChange(i,
e.target.value)
                                                    }
                                                    onKeyDown={(e) => {
                                                        if (
                                                            e.key === "Backspace" &&
                                                            !formData.pin[i] &&
                                                            i > 0
                                                        ) {
                                                            const prevInput =
document.getElementById(`pin-${i - 1}`);
                                                            prevInput?.focus();
                                                        }
                                                    }}
                                                />
                                            ))}
                                        </div>
```

```
                                </div>
                            )}

                            <div className="flex gap-4">
                                <Button
                                    variant="outline"
                                    className="flex-1"
                                    onClick={() => setStep(1)}
                                >
                                    Back
                                </Button>
                                <Button
                                    className="flex-1 bg-[#00B2FF] hover:bg-
[#00B2FF]/90"
                                    onClick={() => setStep(3)}
                                >
                                    Next
                                </Button>
                            </div>
                        </div>
                    )}

                    {step === 3 && (
                        <div className="space-y-6">
                            <h3 className="text-lg font-semibold">Set
Permissions</h3>

                            {Object.entries(formData.permissions).map(([key,
value]) => (
                                <div key={key} className="flex items-center
justify-between">

                                    <Label htmlFor={key} className="flex items-
center space-x-2">
                                        <span>
                                            {key
                                                .replace(/([A-Z])/g, " $1")
                                                .replace(/^./, (str) =>
str.toUpperCase())}
                                        </span>
                                    </Label>
                                    <Switch
                                        id={key}
                                        checked={value}
                                        onCheckedChange={(checked) =>
                                            setFormData({
                                                ...formData,
                                                permissions: {
                                                    ...formData.permissions,
                                                    [key]: checked,
                                                },
                                            })
                                        }
                                        className="data-[state=checked]:bg-
```

```
[#00B2FF]"
                                        />
                                    </div>
                                ))}
                                <div className="flex gap-4">
                                    <Button
                                        variant="outline"
                                        className="flex-1"
                                        onClick={() => setStep(2)}
                                    >
                                        Back
                                    </Button>
                                    <Button
                                        className="flex-1 bg-[#00B2FF] hover:bg-
[#00B2FF]/90"
                                        onClick={handleSubmit}
                                    >
                                        Create Profile
                                    </Button>
                                </div>
                            </div>
                        )}
                    </CardContent>
                </Card>
            </motion.div>
        </div>
    );
}
```

app/dashboard/page.tsx

```
// dashboard/page.tsx
"use client";

import { useState, useEffect } from "react";
import { DndContext, closestCenter, KeyboardSensor, PointerSensor, useSensor,
useSensors } from "@dnd-kit/core";
import { arrayMove, SortableContext, sortableKeyboardCoordinates,
rectSortingStrategy } from "@dnd-kit/sortable";
import { Edit2, Download, Bell, Command, HelpCircle, Sun, Moon, Lock } from "lucide-
react";
import { Button } from "@/components/ui/button";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";
import { ActiveDevices } from "../components/active-devices";
import { EnergyConsumption } from "../components/energy-consumption";
import { FamilyMembers } from "../components/family-members";
import { WeatherWidget } from "../components/weather-widget";
import { SortableWidget } from "../components/sortable-widget";
import { AutomationWidget } from "../components/automation-widget";
import { RoomEnergyConsumption } from "../components/room-energy-consumption";
```

```
import { TimeDateWidget } from "../components/time-date-widget";
import { motion } from "framer-motion";
import { useRouter } from "next/navigation";
import { toast } from "@/components/ui/use-toast";
import { useUser } from "@/contexts/UserContext"; // Import the user context
import { NavigationSidebar } from "@/app/components/navigation-sidebar";
import ProtectedRoute from "@/components/ProtectedRoute"; // Import the protected
route component

const API_URL = process.env.NEXT_PUBLIC_API_URL || 'http://localhost:8000/api/v1';

const initialWidgets = [
  { id: "time-date", component: TimeDateWidget },
  { id: "devices", component: ActiveDevices },
  { id: "weather", component: WeatherWidget },
  { id: "energy", component: EnergyConsumption },
  { id: "automation", component: AutomationWidget },
]

export default function Page() {
  const router = useRouter();
  const { user } = useUser(); // Use the user context

  const [rooms, setRooms] = useState([
    { id: "1", name: "Living Room", consumption: 45 },
    { id: "2", name: "Bedroom", consumption: 18 },
    { id: "3", name: "Kitchen", consumption: 72 },
    { id: "4", name: "Bathroom", consumption: 12 },
    { id: "5", name: "Office", consumption: 35 },
  ]);
  const [isEditing, setIsEditing] = useState(false);
  const [widgets, setWidgets] = useState(initialWidgets);
  const [isDarkMode, setIsDarkMode] = useState(false);
  const [userPermissions, setUserPermissions] = useState({
    statisticalData: true,
    addAutomation: true,
    roomControl: true
  });

  // Use the user data from context instead of making separate API calls
  useEffect(() => {
    // If we have actual permissions endpoints, we can fetch them here
    // For now, we'll use default permissions or mock them based on the user's email
    if (user) {
      // This is a simplified mock - in a real app, you'd fetch real permissions
from your API
      const mockPermissions = {
        statisticalData: true,
        addAutomation: user.email.includes("admin") ? true : false,
        roomControl: true
      };
```

```
      setUserPermissions(mockPermissions);

      // Filter widgets based on permissions
      const filteredWidgets = initialWidgets.filter((widget) => {
        if (widget.id === "energy" && !mockPermissions.statisticalData) return
false;
        if (widget.id === "automation" && !mockPermissions.addAutomation) return
false;
        return true;
      });

      setWidgets(filteredWidgets);
    }
  }, [user]);

  const sensors = useSensors(
    useSensor(PointerSensor),
    useSensor(KeyboardSensor, {
      coordinateGetter: sortableKeyboardCoordinates,
    }),
  );

  function handleDragEnd(event: any) {
    const { active, over } = event;

    if (active.id !== over.id) {
      setWidgets((items) => {
        const oldIndex = items.findIndex((item) => item.id === active.id);
        const newIndex = items.findIndex((item) => item.id === over.id);
        return arrayMove(items, oldIndex, newIndex);
      });
    }
  }

  const containerVariants = {
    hidden: { opacity: 0 },
    visible: {
      opacity: 1,
      transition: {
        staggerChildren: 0.1,
      },
    },
  };

  const itemVariants = {
    hidden: { y: 20, opacity: 0 },
    visible: {
      y: 0,
      opacity: 1,
    },
  };
```

```
  const handleRequestAccess = (feature: string) => {
    // In a real application, this would send a request to the admin
    toast({
      title: "Access Requested",
      description: `Your request for access to ${feature} has been sent to the
admin.`,
    });
  };

  // Wrap the entire dashboard in the ProtectedRoute component
  return (
    <ProtectedRoute>
      <div className={`p-6 min-h-screen ${isDarkMode ? "bg-gray-900 text-white" :
"bg-gray-50 text-gray-900"}`}>
        <NavigationSidebar />
        <div className="ml-[72px]">
          <motion.header variants={itemVariants} className="flex justify-between
items-center mb-6">
            <div className="flex items-center gap-4">
              <div className="w-12 h-12 bg-gradient-to-br from-[#00B2FF] to-
[#0085FF] rounded-full flex items-center justify-center shadow-lg">
                <span className="text-white font-bold text-xl">Sy</span>
              </div>
              <div>
                <h1 className="text-2xl font-bold">Welcome Home,
{user?.email?.split("@")[0] || "User"}</h1>
                <p className="text-sm text-gray-500">Your smart home dashboard</p>
              </div>
            </div>
            <div className="flex items-center space-x-2">
              {userPermissions.addAutomation && (
                <Button
                  variant="ghost"
                  size="icon"
                  onClick={() => setIsEditing(!isEditing)}
                  className={isEditing ? "bg-blue-100" : ""}
                >
                  <Edit2 className="h-5 w-5" />
                </Button>
              )}
              <Button variant="ghost" size="icon">
                <Download className="h-5 w-5" />
              </Button>
              <Button variant="ghost" size="icon">
                <Bell className="h-5 w-5" />
              </Button>
              <Button variant="ghost" size="icon">
                <Command className="h-5 w-5" />
              </Button>
              <Button variant="ghost" size="icon">
                <HelpCircle className="h-5 w-5" />
              </Button>
```

```
              <Button variant="ghost" size="icon" onClick={() =>
setIsDarkMode(!isDarkMode)}>
                {isDarkMode ? <Sun className="h-5 w-5" /> : <Moon className="h-5 w-
5" />}
              </Button>
            </div>
          </motion.header>

          <div className="grid grid-cols-12 gap-6">
            <motion.div variants={itemVariants} className="col-span-12 lg:col-span-
8">
              {userPermissions.roomControl ? (
                <Card className={`mb-6 ${isDarkMode ? "bg-gray-800" : "bg-white"}`}>
                  <CardHeader>
                    <CardTitle>Room Energy Overview</CardTitle>
                  </CardHeader>
                  <CardContent>
                    <RoomEnergyConsumption rooms={rooms} />
                  </CardContent>
                </Card>
              ) : (
                <Card className={`mb-6 ${isDarkMode ? "bg-gray-800" : "bg-white"}`}>
                  <CardContent className="flex flex-col items-center justify-center
p-6">
                    <Lock className="w-12 h-12 text-gray-400 mb-4" />
                    <h3 className="text-lg font-semibold mb-2">Access Required</h3>
                    <p className="text-sm text-gray-500 text-center mb-4">
                      You don't have permission to view room control data.
                    </p>
                    <Button onClick={() => handleRequestAccess("Room
Control")}>Request Access</Button>
                  </CardContent>
                </Card>
              )}
              <DndContext sensors={sensors} collisionDetection={closestCenter}
onDragEnd={handleDragEnd}>
                <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-
4">
                  <SortableContext items={widgets.map((w) => w.id)} strategy=
{rectSortingStrategy}>
                    {widgets.map((widget) => (
                      <SortableWidget key={widget.id} id={widget.id} isEditing=
{isEditing}>
                        <Card className={`h-full ${isDarkMode ? "bg-gray-800" : "bg-
white"}`}>
                          <CardContent className="p-4">
                            {/* Pass user information from the context */}
                            <widget.component
                              userId={user?.id}
                              email={user?.email}
                              token={user?.token}
                            />
```

```
                        </CardContent>
                      </Card>
                    </SortableWidget>
                  ))}
                </SortableContext>
              </div>
            </DndContext>
          </motion.div>
          <motion.div variants={itemVariants} className="col-span-12 lg:col-span-
4">
            <Card className={`h-full ${isDarkMode ? "bg-gray-800" : "bg-white"}`}>
              <CardHeader>
                <CardTitle>Family Members</CardTitle>
              </CardHeader>
              <CardContent>
                <FamilyMembers />
              </CardContent>
            </Card>
          </motion.div>
        </div>
      </div>
    </div>
  </ProtectedRoute>
 );
}
```

app/dashboard/page.tsx.bak

```
"use client";

import { useState, useEffect } from "react";
import { DndContext, closestCenter, KeyboardSensor, PointerSensor, useSensor,
useSensors } from "@dnd-kit/core";
import { arrayMove, SortableContext, sortableKeyboardCoordinates,
rectSortingStrategy } from "@dnd-kit/sortable";
import { Edit2, Download, Bell, Command, HelpCircle, Sun, Moon, Lock } from "lucide-
react";
import { Button } from "@/components/ui/button";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";
import { ActiveDevices } from "../components/active-devices";
import { EnergyConsumption } from "../components/energy-consumption";
import { FamilyMembers } from "../components/family-members";
import { WeatherWidget } from "../components/weather-widget";
import { SortableWidget } from "../components/sortable-widget";
import { AutomationWidget } from "../components/automation-widget";
import { RoomEnergyConsumption } from "../components/room-energy-consumption";
import { TimeDateWidget } from "../components/time-date-widget";
import { motion } from "framer-motion";
import { useRouter } from "next/navigation";
import { toast } from "@/components/ui/use-toast";
```

```
import axios from "axios";
import { NavigationSidebar } from "@/app/components/navigation-sidebar"; // Import
the navbar


const API_URL = process.env.NEXT_PUBLIC_API_URL || 'http://localhost:8000';

const initialWidgets = [
  { id: "time-date", component: TimeDateWidget },
  { id: "devices", component: ActiveDevices },
  { id: "weather", component: WeatherWidget },
  { id: "energy", component: EnergyConsumption },
  { id: "automation", component: AutomationWidget },
]

export default function Page() {
  const router = useRouter()
  const [rooms, setRooms] = useState([
    { id: "1", name: "Living Room", consumption: 45 },
    { id: "2", name: "Bedroom", consumption: 18 },
    { id: "3", name: "Kitchen", consumption: 72 },
    { id: "4", name: "Bathroom", consumption: 12 },
    { id: "5", name: "Office", consumption: 35 },
  ])
  const [isEditing, setIsEditing] = useState(false)
  const [widgets, setWidgets] = useState(initialWidgets)
  const [isDarkMode, setIsDarkMode] = useState(false)
  const [user, setUser] = useState<any>(null)

  useEffect(() => {
    const storedUser = localStorage.getItem("currentUser");
    if (!storedUser) {
      router.push("/auth/login");
      return;
    }

    const currentUser = JSON.parse(storedUser);

    const fetchUserData = async () => {
      try {
        const userResponse = await
axios.get(`${API_URL}/api/user/${currentUser.id}`);
        const userData = userResponse.data;

        const permissionsResponse = await
axios.get(`${API_URL}/api/permissions/${userData.admin_email}`);
        const permissionsData = permissionsResponse.data;

        const updatedUser = { ...userData, ...currentUser, permissions:
permissionsData };
        setUser(updatedUser);
        localStorage.setItem("currentUser", JSON.stringify(updatedUser));
```

```
      } catch (error) {
        console.error("Error fetching user data or permissions:", error);
        toast({
          title: "Error",
          description: "Failed to load user data. Please try again.",
          variant: "destructive",
        });
        router.push("/auth/login");
      }
    };

    fetchUserData();
  }, [router]);

  useEffect(() => {
    if (user) {
      const filteredWidgets = initialWidgets.filter((widget) => {
        if (widget.id === "energy" && !user.permissions?.statisticalData) return
false;
        if (widget.id === "automation" && !user.permissions?.addAutomation) return
false;
        return true;
      });
      setWidgets(filteredWidgets);
    }
  }, [user]);

  const sensors = useSensors(
    useSensor(PointerSensor),
    useSensor(KeyboardSensor, {
      coordinateGetter: sortableKeyboardCoordinates,
    }),
  )

  function handleDragEnd(event: any) {
    const { active, over } = event

    if (active.id !== over.id) {
      setWidgets((items) => {
        const oldIndex = items.findIndex((item) => item.id === active.id)
        const newIndex = items.findIndex((item) => item.id === over.id)
        return arrayMove(items, oldIndex, newIndex)
      })
    }
  }

  const containerVariants = {
    hidden: { opacity: 0 },
    visible: {
      opacity: 1,
      transition: {
```

```
        staggerChildren: 0.1,
      },
    },
  }

  const itemVariants = {
    hidden: { y: 20, opacity: 0 },
    visible: {
      y: 0,
      opacity: 1,
    },
  }

  const handleRequestAccess = (feature: string) => {
    // In a real application, this would send a request to the admin
    toast({
      title: "Access Requested",
      description: `Your request for access to ${feature} has been sent to the
admin.`,
    })
  }

  if (!user) return null

  return (
    <div className={`p-6 min-h-screen bg-gray-50 ${isDarkMode ? "bg-gray-900 text-
white" : "bg-gray-50 text-gray-900"}`}>
      <NavigationSidebar /> {/* Add the navbar here */}
      <div className="ml-[72px]">
        <motion.header variants={itemVariants} className="flex justify-between
items-center mb-6">
          <div className="flex items-center gap-4">
            <div className="w-12 h-12 bg-gradient-to-br from-[#00B2FF] to-[#0085FF]
rounded-full flex items-center justify-center shadow-lg">
              <span className="text-white font-bold text-xl">Sy</span>
            </div>
            <div>
              <h1 className="text-2xl font-bold">Welcome Home, {user.firstName}</h1>
              <p className="text-sm text-gray-500">Your smart home dashboard</p>
            </div>
          </div>
          <div className="flex items-center space-x-2">
            {user.type === "admin" && user.permissions?.addAutomation && (
              <Button
                variant="ghost"
                size="icon"
                onClick={() => setIsEditing(!isEditing)}
                className={isEditing ? "bg-blue-100" : ""}
              >
                <Edit2 className="h-5 w-5" />
              </Button>
            )}
```

```
            <Button variant="ghost" size="icon">
              <Download className="h-5 w-5" />
            </Button>
            <Button variant="ghost" size="icon">
              <Bell className="h-5 w-5" />
            </Button>
            <Button variant="ghost" size="icon">
              <Command className="h-5 w-5" />
            </Button>
            <Button variant="ghost" size="icon">
              <HelpCircle className="h-5 w-5" />
            </Button>
            <Button variant="ghost" size="icon" onClick={() =>
setIsDarkMode(!isDarkMode)}>
              {isDarkMode ? <Sun className="h-5 w-5" /> : <Moon className="h-5 w-5"
/>}
            </Button>
          </div>
        </motion.header>

        <div className="grid grid-cols-12 gap-6">
          <motion.div variants={itemVariants} className="col-span-12 lg:col-span-8">
            {user.permissions?.roomControl ? (
              <Card className={`mb-6 ${isDarkMode ? "bg-gray-800" : "bg-white"}`}>
                <CardHeader>
                  <CardTitle>Room Energy Overview</CardTitle>
                </CardHeader>
                <CardContent>
                  <RoomEnergyConsumption rooms={rooms} />
                </CardContent>
              </Card>
            ) : (
              <Card className={`mb-6 ${isDarkMode ? "bg-gray-800" : "bg-white"}`}>
                <CardContent className="flex flex-col items-center justify-center p-
6">
                  <Lock className="w-12 h-12 text-gray-400 mb-4" />
                  <h3 className="text-lg font-semibold mb-2">Access Required</h3>
                  <p className="text-sm text-gray-500 text-center mb-4">
                    You don't have permission to view room control data.
                  </p>
                  <Button onClick={() => handleRequestAccess("Room
Control")}>Request Access</Button>
                </CardContent>
              </Card>
            )}
            <DndContext sensors={sensors} collisionDetection={closestCenter}
onDragEnd={handleDragEnd}>
              <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4">
                <SortableContext items={widgets.map((w) => w.id)} strategy=
{rectSortingStrategy}>
                  {widgets.map((widget) => (
                    <SortableWidget key={widget.id} id={widget.id} isEditing=
```

```
{isEditing}>
                      <Card className={`h-full ${isDarkMode ? "bg-gray-800" : "bg-
white"}`}>
                        <CardContent className="p-4">
                          {/* Pass user_id or household_id to components needing
user-specific data */}
                          <widget.component userId={user.id} householdId=
{user.household_id} />
                        </CardContent>
                      </Card>
                    </SortableWidget>
                  ))}
                </SortableContext>
              </div>
            </DndContext>
          </motion.div>
          <motion.div variants={itemVariants} className="col-span-12 lg:col-span-4">
            <Card className={`h-full ${isDarkMode ? "bg-gray-800" : "bg-white"}`}>
              <CardHeader>
                <CardTitle>Family Members</CardTitle>
              </CardHeader>
              <CardContent>
                <FamilyMembers />
              </CardContent>
            </Card>
          </motion.div>
        </div>
      </div>
    </div>
  )
}
```

app/dashboard/page.tsx.bak.bak

```
// dashboard/page.tsx
// NOTE: This is a working copy
"use client"

import { useState, useEffect } from 'react';
import { useUser } from '@/contexts/UserContext';
import ProtectedRoute from '@/components/ProtectedRoute';
import { Button } from '@/components/ui/button';

export default function Dashboard() {
  const { user, logout } = useUser();
  const [greeting, setGreeting] = useState('');

  useEffect(() => {
    const hours = new Date().getHours();
    if (hours < 12) {
```

```
      setGreeting('Good morning');
    } else if (hours < 18) {
      setGreeting('Good afternoon');
    } else {
      setGreeting('Good evening');
    }
  }, []);

  return (
    <ProtectedRoute>
      <div className="container mx-auto p-6">
        <div className="flex justify-between items-center mb-8">
          <h1 className="text-3xl font-bold">Dashboard</h1>
          <Button
            variant="outline"
            onClick={logout}
            className="px-4 py-2"
          >
            Logout
          </Button>
        </div>

        <div className="bg-white rounded-lg shadow-md p-6 mb-6">
          <h2 className="text-xl font-semibold mb-4">{greeting}, {user?.email}</h2>
          <p className="text-gray-600">Welcome to your Smart Home Dashboard.</p>
        </div>

        {/* Sample dashboard content */}
        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
          <div className="bg-white rounded-lg shadow-md p-6">
            <h3 className="font-semibold mb-2">Energy Usage</h3>
            <p className="text-gray-600">View your energy consumption statistics</p>
          </div>

          <div className="bg-white rounded-lg shadow-md p-6">
            <h3 className="font-semibold mb-2">Device Management</h3>
            <p className="text-gray-600">Control your connected devices</p>
          </div>

          <div className="bg-white rounded-lg shadow-md p-6">
            <h3 className="font-semibold mb-2">Profiles</h3>
            <p className="text-gray-600">Manage family member profiles</p>
          </div>
        </div>
      </div>
    </ProtectedRoute>
  );
}
```

app/devices/page.tsx

```tsx
// devices/page.tsx
"use client"

import { useState, useEffect } from "react"
import { Card, CardHeader, CardTitle, CardContent } from "@/components/ui/card"
import { Button } from "@/components/ui/button"
import { Switch } from "@/components/ui/switch"
import { Slider } from "@/components/ui/slider"
import { Lightbulb, Thermometer, Fan, Tv, Lock, Plus, Smartphone, Edit, Trash2 }
from "lucide-react"
import Link from "next/link"
import { motion } from "framer-motion"
import { Dialog, DialogContent, DialogHeader, DialogTitle, DialogFooter } from
"@/components/ui/dialog"
import { Label } from "@/components/ui/label"
import { Input } from "@/components/ui/input"
import { NavigationSidebar } from "@/app/components/navigation-sidebar"; // Import
the navbar



interface Device {
  id: string
  name: string
  type: string
  room: string
  status: "on" | "off"
  brightness?: number
  temperature?: number
  speed?: number
  powerConsumption?: number
  totalEnergyConsumed?: number
  lastStatusChange?: number
}

const deviceIcons = {
  light: Lightbulb,
  thermostat: Thermometer,
  fan: Fan,
  tv: Tv,
  lock: Lock,
}

const Icon = ({ type }: { type: string }) => {
  const IconComponent = deviceIcons[type as keyof typeof deviceIcons] || null

  if (!IconComponent) {
    return null;
  }

  return <IconComponent className="h-6 w-6" />
```

```
}

export default function DevicesPage() {
  const [devices, setDevices] = useState<Device[]>([])
  const [editingDevice, setEditingDevice] = useState<Device | null>(null)

  useEffect(() => {
    const storedDevices = JSON.parse(localStorage.getItem("devices") || "[]")
    setDevices(storedDevices)
  }, [])

  const updateDevice = (id: string, updates: Partial<Device>) => {
    const updatedDevices = devices.map((device) => {
      if (device.id === id) {
        const newDevice = { ...device, ...updates }

        if (updates.status !== undefined && updates.status !== device.status) {
          const now = Date.now()
          const timeOn = (now - (device.lastStatusChange || now)) / (1000 * 60 * 60)
// hours
          if (device.status === "on") {
            newDevice.totalEnergyConsumed =
              (newDevice.totalEnergyConsumed || 0) + ((device.powerConsumption || 0)
/ 1000) * timeOn
          }
          newDevice.lastStatusChange = now
        }

        return newDevice
      }
      return device
    })
    setDevices(updatedDevices)
    localStorage.setItem("devices", JSON.stringify(updatedDevices))
  }

  const deleteDevice = (id: string) => {
    if (window.confirm("Are you sure you want to delete this device?")) {
      const updatedDevices = devices.filter((device) => device.id !== id)
      setDevices(updatedDevices)
      localStorage.setItem("devices", JSON.stringify(updatedDevices))
    }
  }

  const handleEditDevice = (device: Device) => {
    setEditingDevice(device)
  }

  const saveEditedDevice = () => {
    if (editingDevice) {
      const updatedDevices = devices.map((device) => (device.id === editingDevice.id
? editingDevice : device))
```

```
      setDevices(updatedDevices)
      localStorage.setItem("devices", JSON.stringify(updatedDevices))
      setEditingDevice(null)
    }
  }

  const renderDeviceControls = (device: Device) => {
    switch (device.type) {
      case "light":
        return (
          <div className="space-y-4">
            <div className="flex items-center justify-between">
              <span>Power</span>
              <Switch
                checked={device.status === "on"}
                onCheckedChange={(checked) => updateDevice(device.id, { status:
checked ? "on" : "off" })}
              />
            </div>
            {device.status === "on" && (
              <div className="space-y-2">
                <span>Brightness</span>
                <Slider
                  value={[device.brightness || 100]}
                  onValueChange={([value]) => updateDevice(device.id, { brightness:
value })}
                  max={100}
                  step={1}
                />
              </div>
            )}
          </div>
        )
      case "thermostat":
        return (
          <div className="space-y-4">
            <div className="flex items-center justify-between">
              <span>Power</span>
              <Switch
                checked={device.status === "on"}
                onCheckedChange={(checked) => updateDevice(device.id, { status:
checked ? "on" : "off" })}
              />
            </div>
            {device.status === "on" && (
              <div className="space-y-2">
                <span>Temperature (°C)</span>
                <Slider
                  value={[device.temperature || 22]}
                  onValueChange={([value]) => updateDevice(device.id, { temperature:
value })}
                  min={16}
```

```
                  max={30}
                  step={0.5}
                />
              </div>
            )}
          </div>
        )
      case "fan":
        return (
          <div className="space-y-4">
            <div className="flex items-center justify-between">
              <span>Power</span>
              <Switch
                checked={device.status === "on"}
                onCheckedChange={(checked) => updateDevice(device.id, { status:
checked ? "on" : "off" })}
              />
            </div>
            {device.status === "on" && (
              <div className="space-y-2">
                <span>Speed</span>
                <Slider
                  value={[device.speed || 1]}
                  onValueChange={([value]) => updateDevice(device.id, { speed: value
})}
                  min={1}
                  max={5}
                  step={1}
                />
              </div>
            )}
          </div>
        )
      default:
        return (
          <div className="flex items-center justify-between">
            <span>Power</span>
            <Switch
              checked={device.status === "on"}
              onCheckedChange={(checked) => updateDevice(device.id, { status:
checked ? "on" : "off" })}
            />
          </div>
        )
    }
  }

  return (
    <div className="min-h-screen bg-gray-50 flex">
      <NavigationSidebar /> {/* Add the navbar here */}
      <div className="flex-1 ml-[72px]">
        <motion.div
```

```
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ duration: 0.5 }}
          className="p-6"
        >
          <motion.header
            initial={{ opacity: 0, y: -20 }}
            animate={{ opacity: 1, y: 0 }}
            transition={{ duration: 0.5 }}
            className="flex justify-between items-center mb-6"
          >
            <div className="flex items-center gap-2">
              <div className="w-12 h-12 bg-gradient-to-br from-[#00B2FF] to-
[#0085FF] rounded-full flex items-center justify-center shadow-lg">
                <Smartphone className="h-6 w-6 text-white" />
              </div>
              <div>
                <h1 className="text-2xl font-bold text-gray-800">Devices</h1>
                <p className="text-sm text-gray-500">Manage and control your smart
devices</p>
              </div>
            </div>
            {devices.length > 0 && (
              <Link href="/add-device">
                <Button className="bg-[#00B2FF] hover:bg-[#00B2FF]/90">
                  <Plus className="mr-2 h-4 w-4" /> Add Device
                </Button>
              </Link>
            )}
          </motion.header>

          <motion.div
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            transition={{ delay: 0.2 }}
            className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4
gap-6"
          >
            {devices.length === 0 ? (
              <Card className="col-span-full p-6 text-center">
                <div className="flex flex-col items-center justify-center space-y-
4">
                  <div className="rounded-full bg-blue-100 p-3">
                    <Smartphone className="h-8 w-8 text-[#00B2FF]" />
                  </div>
                  <div className="space-y-2">
                    <h3 className="font-semibold text-lg">No devices added yet</h3>
                    <p className="text-muted-foreground">Get started by adding your
first smart device</p>
                  </div>
                  <Link href="/add-device">
                    <Button className="bg-[#00B2FF] hover:bg-[#00B2FF]/90">
```

```
                    <Plus className="mr-2 h-4 w-4" /> Add Your First Device
                  </Button>
                </Link>
              </div>
            </Card>
          ) : (
            devices.map((device) => (
              <motion.div
                key={device.id}
                initial={{ opacity: 0, y: 20 }}
                animate={{ opacity: 1, y: 0 }}
                transition={{ duration: 0.3 }}
              >
                <Card className="overflow-hidden">
                  <CardHeader className="bg-gradient-to-r from-[#00B2FF] to-
[#0085FF] text-white p-4">
                    <CardTitle className="flex items-center justify-between">
                      <span className="text-lg font-semibold">{device.name}</span>
                      <Icon type={device.type} />
                    </CardTitle>
                    <p className="text-sm opacity-80">{device.room}</p>
                  </CardHeader>
                  <CardContent className="p-4">
                    <div className="space-y-4">
                      <div className="flex items-center justify-between">
                        <span className="text-sm font-medium text-gray-
500">Status</span>
                        <span
                          className={`text-sm font-medium ${device.status === "on"
? "text-green-500" : "text-red-500"}`}
                        >
                          {device.status === "on" ? "On" : "Off"}
                        </span>
                      </div>
                      {renderDeviceControls(device)}
                      {device.powerConsumption && (
                        <div className="flex items-center justify-between text-
sm">
                          <span className="text-gray-500">Power Consumption</span>
                          <span className="font-medium">{device.powerConsumption}
W</span>
                        </div>
                      )}
                      {device.totalEnergyConsumed && (
                        <div className="flex items-center justify-between text-
sm">
                          <span className="text-gray-500">Total Energy
Consumed</span>
                          <span className="font-medium">
{device.totalEnergyConsumed.toFixed(2)} kWh</span>
                        </div>
                      )}
```

```
                        <div className="flex justify-end space-x-2 mt-4">
                          <Button variant="outline" size="sm" onClick={() =>
handleEditDevice(device)}>
                            <Edit className="w-4 h-4" />
                          </Button>
                          <Button variant="outline" size="sm" onClick={() =>
deleteDevice(device.id)}>
                            <Trash2 className="w-4 h-4" />
                          </Button>
                        </div>
                      </div>
                    </CardContent>
                  </Card>
                </motion.div>
              ))
            )}
          </motion.div>

        </motion.div>
      </div>
    </div>
  )
}
```

app/devices/page.tsx.bak

```
"use client"

import { useState, useEffect } from "react"
import { Card, CardHeader, CardTitle, CardContent } from "@/components/ui/card"
import { Button } from "@/components/ui/button"
import { Switch } from "@/components/ui/switch"
import { Slider } from "@/components/ui/slider"
import { Lightbulb, Thermometer, Fan, Tv, Lock, Plus, Smartphone, Edit, Trash2 }
from "lucide-react"
import Link from "next/link"
import { motion } from "framer-motion"
import { Dialog, DialogContent, DialogHeader, DialogTitle, DialogFooter } from
"@/components/ui/dialog"
import { Label } from "@/components/ui/label"
import { Input } from "@/components/ui/input"
import { NavigationSidebar } from "@/app/components/navigation-sidebar"; // Import
the navbar



interface Device {
  id: string
  name: string
  type: string
```

```
  room: string
  status: "on" | "off"
  brightness?: number
  temperature?: number
  speed?: number
  powerConsumption?: number
  totalEnergyConsumed?: number
  lastStatusChange?: number
}

const deviceIcons = {
  light: Lightbulb,
  thermostat: Thermometer,
  fan: Fan,
  tv: Tv,
  lock: Lock,
}

const Icon = ({ type }: { type: string }) => {
  const IconComponent = deviceIcons[type as keyof typeof deviceIcons] || null
  return <IconComponent className="h-6 w-6" />
}

export default function DevicesPage() {
  const [devices, setDevices] = useState<Device[]>([])
  const [editingDevice, setEditingDevice] = useState<Device | null>(null)

  useEffect(() => {
    const storedDevices = JSON.parse(localStorage.getItem("devices") || "[]")
    setDevices(storedDevices)
  }, [])

  const updateDevice = (id: string, updates: Partial<Device>) => {
    const updatedDevices = devices.map((device) => {
      if (device.id === id) {
        const newDevice = { ...device, ...updates }

        if (updates.status !== undefined && updates.status !== device.status) {
          const now = Date.now()
          const timeOn = (now - (device.lastStatusChange || now)) / (1000 * 60 * 60)
// hours
          if (device.status === "on") {
            newDevice.totalEnergyConsumed =
              (newDevice.totalEnergyConsumed || 0) + ((device.powerConsumption || 0)
/ 1000) * timeOn
          }
          newDevice.lastStatusChange = now
        }

        return newDevice
      }
      return device
```

```
    })
    setDevices(updatedDevices)
    localStorage.setItem("devices", JSON.stringify(updatedDevices))
  }

  const deleteDevice = (id: string) => {
    if (window.confirm("Are you sure you want to delete this device?")) {
      const updatedDevices = devices.filter((device) => device.id !== id)
      setDevices(updatedDevices)
      localStorage.setItem("devices", JSON.stringify(updatedDevices))
    }
  }

  const handleEditDevice = (device: Device) => {
    setEditingDevice(device)
  }

  const saveEditedDevice = () => {
    if (editingDevice) {
      const updatedDevices = devices.map((device) => (device.id === editingDevice.id
? editingDevice : device))
      setDevices(updatedDevices)
      localStorage.setItem("devices", JSON.stringify(updatedDevices))
      setEditingDevice(null)
    }
  }

  const renderDeviceControls = (device: Device) => {
    switch (device.type) {
      case "light":
        return (
          <div className="space-y-4">
            <div className="flex items-center justify-between">
              <span>Power</span>
              <Switch
                checked={device.status === "on"}
                onCheckedChange={(checked) => updateDevice(device.id, { status:
checked ? "on" : "off" })}
              />
            </div>
            {device.status === "on" && (
              <div className="space-y-2">
                <span>Brightness</span>
                <Slider
                  value={[device.brightness || 100]}
                  onValueChange={([value]) => updateDevice(device.id, { brightness:
value })}
                  max={100}
                  step={1}
                />
              </div>
            )}
```

```
          </div>
        )
      case "thermostat":
        return (
          <div className="space-y-4">
            <div className="flex items-center justify-between">
              <span>Power</span>
              <Switch
                checked={device.status === "on"}
                onCheckedChange={(checked) => updateDevice(device.id, { status:
checked ? "on" : "off" })}
              />
            </div>
            {device.status === "on" && (
              <div className="space-y-2">
                <span>Temperature (°C)</span>
                <Slider
                  value={[device.temperature || 22]}
                  onValueChange={([value]) => updateDevice(device.id, { temperature:
value })}
                  min={16}
                  max={30}
                  step={0.5}
                />
              </div>
            )}
          </div>
        )
      case "fan":
        return (
          <div className="space-y-4">
            <div className="flex items-center justify-between">
              <span>Power</span>
              <Switch
                checked={device.status === "on"}
                onCheckedChange={(checked) => updateDevice(device.id, { status:
checked ? "on" : "off" })}
              />
            </div>
            {device.status === "on" && (
              <div className="space-y-2">
                <span>Speed</span>
                <Slider
                  value={[device.speed || 1]}
                  onValueChange={([value]) => updateDevice(device.id, { speed: value
})}
                  min={1}
                  max={5}
                  step={1}
                />
              </div>
            )}
```

```
          </div>
        )
      default:
        return (
          <div className="flex items-center justify-between">
            <span>Power</span>
            <Switch
              checked={device.status === "on"}
              onCheckedChange={(checked) => updateDevice(device.id, { status:
checked ? "on" : "off" })}
            />
          </div>
        )
    }
  }

  return (
    <div className="min-h-screen bg-gray-50 flex">
      <NavigationSidebar /> {/* Add the navbar here */}
      <div className="flex-1 ml-[72px]">
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ duration: 0.5 }}
          className="p-6"
        >
          <motion.header
            initial={{ opacity: 0, y: -20 }}
            animate={{ opacity: 1, y: 0 }}
            transition={{ duration: 0.5 }}
            className="flex justify-between items-center mb-6"
          >
            <div className="flex items-center gap-2">
              <div className="w-12 h-12 bg-gradient-to-br from-[#00B2FF] to-
[#0085FF] rounded-full flex items-center justify-center shadow-lg">
                <Smartphone className="h-6 w-6 text-white" />
              </div>
              <div>
                <h1 className="text-2xl font-bold text-gray-800">Devices</h1>
                <p className="text-sm text-gray-500">Manage and control your smart
devices</p>
              </div>
            </div>
            {devices.length > 0 && (
              <Link href="/add-device">
                <Button className="bg-[#00B2FF] hover:bg-[#00B2FF]/90">
                  <Plus className="mr-2 h-4 w-4" /> Add Device
                </Button>
              </Link>
            )}
          </motion.header>
```

```
          <motion.div
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            transition={{ delay: 0.2 }}
            className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4
gap-6"
          >
            {devices.length === 0 ? (
              <Card className="col-span-full p-6 text-center">
                <div className="flex flex-col items-center justify-center space-y-
4">
                  <div className="rounded-full bg-blue-100 p-3">
                    <Smartphone className="h-8 w-8 text-[#00B2FF]" />
                  </div>
                  <div className="space-y-2">
                    <h3 className="font-semibold text-lg">No devices added yet</h3>
                    <p className="text-muted-foreground">Get started by adding your
first smart device</p>
                  </div>
                  <Link href="/add-device">
                    <Button className="bg-[#00B2FF] hover:bg-[#00B2FF]/90">
                      <Plus className="mr-2 h-4 w-4" /> Add Your First Device
                    </Button>
                  </Link>
                </div>
              </Card>
            ) : (
              devices.map((device) => (
                <motion.div
                  key={device.id}
                  initial={{ opacity: 0, y: 20 }}
                  animate={{ opacity: 1, y: 0 }}
                  transition={{ duration: 0.3 }}
                >
                  <Card className="overflow-hidden">
                    <CardHeader className="bg-gradient-to-r from-[#00B2FF] to-
[#0085FF] text-white p-4">
                      <CardTitle className="flex items-center justify-between">
                        <span className="text-lg font-semibold">{device.name}</span>
                        <Icon type={device.type} />
                      </CardTitle>
                      <p className="text-sm opacity-80">{device.room}</p>
                    </CardHeader>
                    <CardContent className="p-4">
                      <div className="space-y-4">
                        <div className="flex items-center justify-between">
                          <span className="text-sm font-medium text-gray-
500">Status</span>
                          <span
                            className={`text-sm font-medium ${device.status === "on"
? "text-green-500" : "text-red-500"}`}
                          >
```

```
                            {device.status === "on" ? "On" : "Off"}
                          </span>
                        </div>
                        {renderDeviceControls(device)}
                        {device.powerConsumption && (
                          <div className="flex items-center justify-between text-
sm">
                            <span className="text-gray-500">Power Consumption</span>
                            <span className="font-medium">{device.powerConsumption}
W</span>
                          </div>
                        )}
                        {device.totalEnergyConsumed && (
                          <div className="flex items-center justify-between text-
sm">
                            <span className="text-gray-500">Total Energy
Consumed</span>
                            <span className="font-medium">
{device.totalEnergyConsumed.toFixed(2)} kWh</span>
                          </div>
                        )}
                        <div className="flex justify-end space-x-2 mt-4">
                          <Button variant="outline" size="sm" onClick={() =>
handleEditDevice(device)}>
                            <Edit className="w-4 h-4" />
                          </Button>
                          <Button variant="outline" size="sm" onClick={() =>
deleteDevice(device.id)}>
                            <Trash2 className="w-4 h-4" />
                          </Button>
                        </div>
                      </div>
                    </CardContent>
                  </Card>
                </motion.div>
              ))
            )}
          </motion.div>

        </motion.div>
      </div>
    </div>
  )
}
```

app/layout.tsx

```
import type { Metadata } from "next";
import { Inter } from "next/font/google";
import "./globals.css";
```

```tsx
import type React from "react";
import { Toaster } from "@/components/ui/toaster";
import { UserProvider } from "@/contexts/UserContext";

const inter = Inter({ subsets: ["latin"] });

export const metadata: Metadata = {
  title: "Smart Home Dashboard",
  description: "Smart home control and monitoring system",
  generator: "Sync Studios",
};

export default function RootLayout({ children }: { children: React.ReactNode }) {
  return (
    <html lang="en">
      <head>
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
      </head>
      <body className={inter.className}>
        <UserProvider>
          {children}
          <Toaster />
        </UserProvider>
      </body>
    </html>
  );
}
```

app/layout.tsx.bak

```tsx
import type { Metadata } from "next";
import { Inter } from "next/font/google";
import "./globals.css";
import type React from "react";
import { Toaster } from "@/components/ui/toaster";

const inter = Inter({ subsets: ["latin"] });

export const metadata: Metadata = {
  title: "Smart Home Dashboard",
  description: "Smart home control and monitoring system",
  generator: "Sync Studios",
};

export default function RootLayout({ children }: { children: React.ReactNode }) {
  return (
    <html lang="en">
      <head>
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
      </head>
```

```
      <body className={inter.className}>
        {children}
        <Toaster />
      </body>
    </html>
  );
}
```

app/manage-profiles/[id]/access/page.tsx

```
// manage-profiles/[id]/access/page.tsx
"use client"

import { useState, useEffect } from "react"
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"
import { Button } from "@/components/ui/button"
import { Switch } from "@/components/ui/switch"
import { motion } from "framer-motion"
import { ArrowLeft, Bell, Zap, Clock, BarChart2, Database, History, Smartphone }
from "lucide-react"
import { useRouter } from "next/navigation"

interface FamilyMember {
  id: string
  name: string
  email: string
  role: string
  permissions?: {
    notifications: boolean
    energyAlerts: boolean
    addAutomation: boolean
    realTimeMonitoring: boolean
    analyticalData: boolean
    historicalData: boolean
    deviceControl: boolean
  }
}

export default function ManageProfileAccessPage({ params }: { params: { id: string }
}) {
  const router = useRouter()
  const [member, setMember] = useState<FamilyMember | null>(null)
  const [permissions, setPermissions] = useState({
    notifications: true,
    energyAlerts: true,
    addAutomation: false,
    realTimeMonitoring: true,
    analyticalData: false,
    historicalData: false,
    deviceControl: true,
```

```
  })

  useEffect(() => {
    const storedMembers = JSON.parse(localStorage.getItem("familyMembers") || "[]")
    const member = storedMembers.find((m: FamilyMember) => m.id === params.id)
    if (member) {
      setMember(member)
      if (member.permissions) {
        setPermissions(member.permissions)
      }
    }
  }, [params.id])

  const handleSaveChanges = () => {
    const storedMembers = JSON.parse(localStorage.getItem("familyMembers") || "[]")
    const updatedMembers = storedMembers.map((m: FamilyMember) => {
      if (m.id === params.id) {
        return {
          ...m,
          permissions,
        }
      }
      return m
    })
    localStorage.setItem("familyMembers", JSON.stringify(updatedMembers))
    router.push("/manage-profiles")
  }

  if (!member) return null

  return (
    <div className="p-6 min-h-screen bg-gradient-to-br from-gray-50 to-gray-100">
      <motion.div initial={{ opacity: 0, y: -20 }} animate={{ opacity: 1, y: 0 }}
transition={{ duration: 0.5 }}>
        <header className="flex justify-between items-center mb-8">
          <div className="flex items-center gap-4">
            <Button variant="ghost" size="icon" onClick={() => router.back()}>
              <ArrowLeft className="h-6 w-6" />
            </Button>
            <div>
              <h1 className="text-2xl font-bold text-gray-800">Manage Access</h1>
              <p className="text-sm text-gray-500">
                {member.name} • {member.role}
              </p>
            </div>
          </div>
        </header>
      </motion.div>

      <div className="grid gap-6 max-w-2xl mx-auto">
        <motion.div initial={{ opacity: 0, y: 20 }} animate={{ opacity: 1, y: 0 }}
transition={{ duration: 0.5 }}>
```

```
          <Card>
            <CardHeader>
              <CardTitle>Access Permissions</CardTitle>
            </CardHeader>
            <CardContent className="space-y-6">
              <div className="flex items-center justify-between">
                <div className="flex items-center gap-3">
                  <Bell className="h-5 w-5 text-[#00B2FF]" />
                  <div>
                    <p className="font-medium">Notifications</p>
                    <p className="text-sm text-gray-500">Receive system
notifications</p>
                  </div>
                </div>
                <Switch
                  checked={permissions.notifications}
                  onCheckedChange={(checked) => setPermissions({ ...permissions,
notifications: checked })}
                  className="data-[state=checked]:bg-[#00B2FF]"
                />
              </div>

              <div className="flex items-center justify-between">
                <div className="flex items-center gap-3">
                  <Zap className="h-5 w-5 text-[#00B2FF]" />
                  <div>
                    <p className="font-medium">Energy Alerts</p>
                    <p className="text-sm text-gray-500">Receive energy consumption
alerts</p>
                  </div>
                </div>
                <Switch
                  checked={permissions.energyAlerts}
                  onCheckedChange={(checked) => setPermissions({ ...permissions,
energyAlerts: checked })}
                  className="data-[state=checked]:bg-[#00B2FF]"
                />
              </div>

              <div className="flex items-center justify-between">
                <div className="flex items-center gap-3">
                  <Clock className="h-5 w-5 text-[#00B2FF]" />
                  <div>
                    <p className="font-medium">Add Automation</p>
                    <p className="text-sm text-gray-500">Create and modify
automations</p>
                  </div>
                </div>
                <Switch
                  checked={permissions.addAutomation}
                  onCheckedChange={(checked) => setPermissions({ ...permissions,
addAutomation: checked })}
```

```
                  className="data-[state=checked]:bg-[#00B2FF]"
                />
              </div>

              <div className="flex items-center justify-between">
                <div className="flex items-center gap-3">
                  <BarChart2 className="h-5 w-5 text-[#00B2FF]" />
                  <div>
                    <p className="font-medium">Real Time Monitoring</p>
                    <p className="text-sm text-gray-500">View live device status</p>
                  </div>
                </div>
                <Switch
                  checked={permissions.realTimeMonitoring}
                  onCheckedChange={(checked) => setPermissions({ ...permissions,
realTimeMonitoring: checked })}
                  className="data-[state=checked]:bg-[#00B2FF]"
                />
              </div>

              <div className="flex items-center justify-between">
                <div className="flex items-center gap-3">
                  <Database className="h-5 w-5 text-[#00B2FF]" />
                  <div>
                    <p className="font-medium">Analytical Data</p>
                    <p className="text-sm text-gray-500">Access usage analytics</p>
                  </div>
                </div>
                <Switch
                  checked={permissions.analyticalData}
                  onCheckedChange={(checked) => setPermissions({ ...permissions,
analyticalData: checked })}
                  className="data-[state=checked]:bg-[#00B2FF]"
                />
              </div>

              <div className="flex items-center justify-between">
                <div className="flex items-center gap-3">
                  <History className="h-5 w-5 text-[#00B2FF]" />
                  <div>
                    <p className="font-medium">Historical Data</p>
                    <p className="text-sm text-gray-500">View past usage data</p>
                  </div>
                </div>
                <Switch
                  checked={permissions.historicalData}
                  onCheckedChange={(checked) => setPermissions({ ...permissions,
historicalData: checked })}
                  className="data-[state=checked]:bg-[#00B2FF]"
                />
              </div>
```

```
              <div className="flex items-center justify-between">
                <div className="flex items-center gap-3">
                  <Smartphone className="h-5 w-5 text-[#00B2FF]" />
                  <div>
                    <p className="font-medium">Device Control</p>
                    <p className="text-sm text-gray-500">Control smart devices</p>
                  </div>
                </div>
                <Switch
                  checked={permissions.deviceControl}
                  onCheckedChange={(checked) => setPermissions({ ...permissions,
deviceControl: checked })}
                  className="data-[state=checked]:bg-[#00B2FF]"
                />
              </div>
            </CardContent>
          </Card>
        </motion.div>

        <motion.div initial={{ opacity: 0, y: 20 }} animate={{ opacity: 1, y: 0 }}
transition={{ delay: 0.2 }}>
          <Button className="w-full bg-[#00B2FF] hover:bg-[#00B2FF]/90" onClick=
{handleSaveChanges}>
            Save Changes
          </Button>
        </motion.div>
      </div>
    </div>
  )
}
```

app/manage-profiles/[id]/details/page.tsx

```
// manage-profiles/[id]/details/page.tsx
"use client"

import { useState, useEffect } from "react"
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"
import { Button } from "@/components/ui/button"
import { Input } from "@/components/ui/input"
import { Label } from "@/components/ui/label"
import { motion } from "framer-motion"
import { ArrowLeft, User, Key, Trash2, Mail } from "lucide-react"
import { useRouter } from "next/navigation"
import {
  Dialog,
  DialogContent,
  DialogDescription,
  DialogFooter,
```

```
  DialogHeader,
  DialogTitle,
  DialogTrigger,
} from "@/components/ui/dialog"
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from
"@/components/ui/select"

interface FamilyMember {
  id: string
  name: string
  email: string
  role: string
  pin?: string
}

export default function ManageProfileDetailsPage({ params }: { params: { id: string
} }) {
  const router = useRouter()
  const [member, setMember] = useState<FamilyMember | null>(null)
  const [isChangePinOpen, setIsChangePinOpen] = useState(false)
  const [newPin, setNewPin] = useState(["", "", "", ""])
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    role: "",
  })

  useEffect(() => {
    const storedMembers = JSON.parse(localStorage.getItem("familyMembers") || "[]")
    const member = storedMembers.find((m: FamilyMember) => m.id === params.id)
    if (member) {
      setMember(member)
      setFormData({
        name: member.name,
        email: member.email,
        role: member.role,
      })
    }
  }, [params.id])

  const handlePinChange = (index: number, value: string) => {
    if (value.length > 1) return
    const newPinArray = [...newPin]
    newPinArray[index] = value
    setNewPin(newPinArray)

    if (value !== "" && index < 3) {
      const nextInput = document.getElementById(`pin-${index + 1}`)
      nextInput?.focus()
    }
  }
```

```
  const handleSaveChanges = () => {
    const storedMembers = JSON.parse(localStorage.getItem("familyMembers") || "[]")
    const updatedMembers = storedMembers.map((m: FamilyMember) => {
      if (m.id === params.id) {
        return {
          ...m,
          ...formData,
        }
      }
      return m
    })
    localStorage.setItem("familyMembers", JSON.stringify(updatedMembers))
    router.push("/manage-profiles")
  }

  const handleSavePin = () => {
    const pin = newPin.join("")
    if (pin.length === 4) {
      const storedMembers = JSON.parse(localStorage.getItem("familyMembers") || "
[]")
      const updatedMembers = storedMembers.map((m: FamilyMember) => {
        if (m.id === params.id) {
          return {
            ...m,
            pin,
          }
        }
        return m
      })
      localStorage.setItem("familyMembers", JSON.stringify(updatedMembers))
      setIsChangePinOpen(false)
      setNewPin(["", "", "", ""])
    }
  }

  const handleDeleteProfile = () => {
    const storedMembers = JSON.parse(localStorage.getItem("familyMembers") || "[]")
    const updatedMembers = storedMembers.filter((m: FamilyMember) => m.id !==
params.id)
    localStorage.setItem("familyMembers", JSON.stringify(updatedMembers))
    router.push("/manage-profiles")
  }

  if (!member) return null

  return (
    <div className="p-6 min-h-screen bg-gradient-to-br from-gray-50 to-gray-100">
      <motion.div initial={{ opacity: 0, y: -20 }} animate={{ opacity: 1, y: 0 }}
transition={{ duration: 0.5 }}>
        <header className="flex justify-between items-center mb-8">
          <div className="flex items-center gap-4">
            <Button variant="ghost" size="icon" onClick={() => router.back()}>
```

```
              <ArrowLeft className="h-6 w-6" />
            </Button>
            <div>
              <h1 className="text-2xl font-bold text-gray-800">Manage Details</h1>
              <p className="text-sm text-gray-500">
                {member.name} • {member.role}
              </p>
            </div>
          </div>
        </header>
      </motion.div>

      <div className="grid gap-6 max-w-2xl mx-auto">
        <motion.div initial={{ opacity: 0, y: 20 }} animate={{ opacity: 1, y: 0 }}
transition={{ duration: 0.5 }}>
          <Card>
            <CardHeader>
              <CardTitle>Profile Information</CardTitle>
            </CardHeader>
            <CardContent className="space-y-6">
              <div className="space-y-2">
                <Label>Name</Label>
                <div className="flex gap-2">
                  <div className="relative flex-1">
                    <Input
                      value={formData.name}
                      onChange={(e) => setFormData({ ...formData, name:
e.target.value })}
                      className="pl-10"
                    />
                    <User className="absolute left-3 top-1/2 -translate-y-1/2 h-5 w-
5 text-gray-400" />
                  </div>
                </div>
              </div>

              <div className="space-y-2">
                <Label>Email</Label>
                <div className="relative">
                  <Input
                    value={formData.email}
                    onChange={(e) => setFormData({ ...formData, email:
e.target.value })}
                    className="pl-10"
                  />
                  <Mail className="absolute left-3 top-1/2 -translate-y-1/2 h-5 w-5
text-gray-400" />
                </div>
              </div>

              <div className="space-y-2">
                <Label>Role</Label>
```

```
                <Select value={formData.role} onValueChange={(value) =>
setFormData({ ...formData, role: value })}>
                  <SelectTrigger>
                    <SelectValue />
                  </SelectTrigger>
                  <SelectContent>
                    <SelectItem value="adult">Adult</SelectItem>
                    <SelectItem value="child">Child</SelectItem>
                    <SelectItem value="admin">Admin</SelectItem>
                  </SelectContent>
                </Select>
              </div>
            </CardContent>
          </Card>
        </motion.div>

        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ delay: 0.2 }}
          className="flex gap-4"
        >
          <Dialog open={isChangePinOpen} onOpenChange={setIsChangePinOpen}>
            <DialogTrigger asChild>
              <Button variant="outline" className="flex-1">
                <Key className="w-4 h-4 mr-2" />
                Change PIN
              </Button>
            </DialogTrigger>
            <DialogContent>
              <DialogHeader>
                <DialogTitle>Change PIN</DialogTitle>
                <DialogDescription>Enter a new 4-digit PIN for this profile.
</DialogDescription>
              </DialogHeader>
              <div className="flex justify-center gap-4 py-4">
                {[0, 1, 2, 3].map((i) => (
                  <Input
                    key={i}
                    id={`pin-${i}`}
                    type="text"
                    inputMode="numeric"
                    pattern="[0-9]*"
                    maxLength={1}
                    className="w-16 h-16 text-2xl text-center"
                    value={newPin[i]}
                    onChange={(e) => handlePinChange(i, e.target.value)}
                    onKeyDown={(e) => {
                      if (e.key === "Backspace" && !newPin[i] && i > 0) {
                        const prevInput = document.getElementById(`pin-${i - 1}`)
                        prevInput?.focus()
                      }
```

```
                  }}
                />
              ))}
            </div>
            <DialogFooter>
              <Button
                onClick={handleSavePin}
                disabled={newPin.some((digit) => !digit)}
                className="w-full bg-[#00B2FF] hover:bg-[#00B2FF]/90"
              >
                Save PIN
              </Button>
            </DialogFooter>
          </DialogContent>
        </Dialog>

        <Button variant="destructive" className="flex-1" onClick=
{handleDeleteProfile}>
          <Trash2 className="w-4 h-4 mr-2" />
          Delete Profile
        </Button>

        <Button className="flex-1 bg-[#00B2FF] hover:bg-[#00B2FF]/90" onClick=
{handleSaveChanges}>
          Save Changes
        </Button>
      </motion.div>
    </div>
  </div>
 )
}
```

app/manage-profiles/[id]/page.tsx

```
// manage-profiles/[id]/page.tsx
"use client"

import { useState, useEffect } from "react"
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"
import { Button } from "@/components/ui/button"
import { Switch } from "@/components/ui/switch"
import { Input } from "@/components/ui/input"
import { motion } from "framer-motion"
import { ArrowLeft, Bell, Zap, Clock, BarChart2, Database, History, Smartphone,
Trash2, Key } from "lucide-react"
import { useRouter } from "next/navigation"
import {
  Dialog,
  DialogContent,
```

```
  DialogDescription,
  DialogFooter,
  DialogHeader,
  DialogTitle,
  DialogTrigger,
} from "@/components/ui/dialog"

interface FamilyMember {
  id: string
  name: string
  email: string
  role: string
  permissions?: {
    notifications: boolean
    energyAlerts: boolean
    addAutomation: boolean
    realTimeMonitoring: boolean
    analyticalData: boolean
    historicalData: boolean
    deviceControl: boolean
  }
  pin?: string
}

export default function ManageProfilePage({ params }: { params: { id: string } }) {
  const router = useRouter()
  const [member, setMember] = useState<FamilyMember | null>(null)
  const [permissions, setPermissions] = useState({
    notifications: true,
    energyAlerts: true,
    addAutomation: false,
    realTimeMonitoring: true,
    analyticalData: false,
    historicalData: false,
    deviceControl: true,
  })
  const [isChangePinOpen, setIsChangePinOpen] = useState(false)
  const [newPin, setNewPin] = useState(["", "", "", ""])

  useEffect(() => {
    const storedMembers = JSON.parse(localStorage.getItem("familyMembers") || "[]")
    const member = storedMembers.find((m: FamilyMember) => m.id === params.id)
    if (member) {
      setMember(member)
      if (member.permissions) {
        setPermissions(member.permissions)
      }
    }
  }, [params.id])

  const handlePinChange = (index: number, value: string) => {
    if (value.length > 1) return
```

```
    const newPinArray = [...newPin]
    newPinArray[index] = value
    setNewPin(newPinArray)

    if (value !== "" && index < 3) {
      const nextInput = document.getElementById(`pin-${index + 1}`)
      nextInput?.focus()
    }
  }

  const handleSaveChanges = () => {
    const storedMembers = JSON.parse(localStorage.getItem("familyMembers") || "[]")
    const updatedMembers = storedMembers.map((m: FamilyMember) => {
      if (m.id === params.id) {
        return {
          ...m,
          permissions,
        }
      }
      return m
    })
    localStorage.setItem("familyMembers", JSON.stringify(updatedMembers))
    router.push("/manage-profiles")
  }

  const handleSavePin = () => {
    const pin = newPin.join("")
    if (pin.length === 4) {
      const storedMembers = JSON.parse(localStorage.getItem("familyMembers") || "
[]")
      const updatedMembers = storedMembers.map((m: FamilyMember) => {
        if (m.id === params.id) {
          return {
            ...m,
            pin,
          }
        }
        return m
      })
      localStorage.setItem("familyMembers", JSON.stringify(updatedMembers))
      setIsChangePinOpen(false)
      setNewPin(["", "", "", ""])
    }
  }

  const handleDeleteProfile = () => {
    const storedMembers = JSON.parse(localStorage.getItem("familyMembers") || "[]")
    const updatedMembers = storedMembers.filter((m: FamilyMember) => m.id !==
params.id)
    localStorage.setItem("familyMembers", JSON.stringify(updatedMembers))
    router.push("/manage-profiles")
  }
```

```
  if (!member) return null

  return (
    <div className="p-6 min-h-screen bg-gradient-to-br from-gray-50 to-gray-100">
      <motion.div initial={{ opacity: 0, y: -20 }} animate={{ opacity: 1, y: 0 }}
transition={{ duration: 0.5 }}>
        <header className="flex justify-between items-center mb-8">
          <div className="flex items-center gap-4">
            <Button variant="ghost" size="icon" onClick={() => router.back()}>
              <ArrowLeft className="h-6 w-6" />
            </Button>
            <div>
              <h1 className="text-2xl font-bold text-gray-800">Manage Access</h1>
              <p className="text-sm text-gray-500">
                {member.name} • {member.role}
              </p>
            </div>
          </div>
        </header>
      </motion.div>

      <div className="grid gap-6">
        <motion.div initial={{ opacity: 0, y: 20 }} animate={{ opacity: 1, y: 0 }}
transition={{ duration: 0.5 }}>
          <Card>
            <CardHeader>
              <CardTitle>Access Permissions</CardTitle>
            </CardHeader>
            <CardContent className="space-y-6">
              <div className="flex items-center justify-between">
                <div className="flex items-center gap-3">
                  <Bell className="h-5 w-5 text-[#00B2FF]" />
                  <div>
                    <p className="font-medium">Notifications</p>
                    <p className="text-sm text-gray-500">Receive system
notifications</p>
                  </div>
                </div>
                <Switch
                  checked={permissions.notifications}
                  onCheckedChange={(checked) => setPermissions({ ...permissions,
notifications: checked })}
                  className="data-[state=checked]:bg-[#00B2FF]"
                />
              </div>

              <div className="flex items-center justify-between">
                <div className="flex items-center gap-3">
                  <Zap className="h-5 w-5 text-[#00B2FF]" />
                  <div>
                    <p className="font-medium">Energy Alerts</p>
```

```
                    <p className="text-sm text-gray-500">Receive energy consumption
alerts</p>
                  </div>
                </div>
                <Switch
                  checked={permissions.energyAlerts}
                  onCheckedChange={(checked) => setPermissions({ ...permissions,
energyAlerts: checked })}
                  className="data-[state=checked]:bg-[#00B2FF]"
                />
              </div>

              <div className="flex items-center justify-between">
                <div className="flex items-center gap-3">
                  <Clock className="h-5 w-5 text-[#00B2FF]" />
                  <div>
                    <p className="font-medium">Add Automation</p>
                    <p className="text-sm text-gray-500">Create and modify
automations</p>
                  </div>
                </div>
                <Switch
                  checked={permissions.addAutomation}
                  onCheckedChange={(checked) => setPermissions({ ...permissions,
addAutomation: checked })}
                  className="data-[state=checked]:bg-[#00B2FF]"
                />
              </div>

              <div className="flex items-center justify-between">
                <div className="flex items-center gap-3">
                  <BarChart2 className="h-5 w-5 text-[#00B2FF]" />
                  <div>
                    <p className="font-medium">Real Time Monitoring</p>
                    <p className="text-sm text-gray-500">View live device status</p>
                  </div>
                </div>
                <Switch
                  checked={permissions.realTimeMonitoring}
                  onCheckedChange={(checked) => setPermissions({ ...permissions,
realTimeMonitoring: checked })}
                  className="data-[state=checked]:bg-[#00B2FF]"
                />
              </div>

              <div className="flex items-center justify-between">
                <div className="flex items-center gap-3">
                  <Database className="h-5 w-5 text-[#00B2FF]" />
                  <div>
                    <p className="font-medium">Analytical Data</p>
                    <p className="text-sm text-gray-500">Access usage analytics</p>
                  </div>
```

```
                </div>
                <Switch
                  checked={permissions.analyticalData}
                  onCheckedChange={(checked) => setPermissions({ ...permissions,
analyticalData: checked })}
                  className="data-[state=checked]:bg-[#00B2FF]"
                />
              </div>

              <div className="flex items-center justify-between">
                <div className="flex items-center gap-3">
                  <History className="h-5 w-5 text-[#00B2FF]" />
                  <div>
                    <p className="font-medium">Historical Data</p>
                    <p className="text-sm text-gray-500">View past usage data</p>
                  </div>
                </div>
                <Switch
                  checked={permissions.historicalData}
                  onCheckedChange={(checked) => setPermissions({ ...permissions,
historicalData: checked })}
                  className="data-[state=checked]:bg-[#00B2FF]"
                />
              </div>

              <div className="flex items-center justify-between">
                <div className="flex items-center gap-3">
                  <Smartphone className="h-5 w-5 text-[#00B2FF]" />
                  <div>
                    <p className="font-medium">Device Control</p>
                    <p className="text-sm text-gray-500">Control smart devices</p>
                  </div>
                </div>
                <Switch
                  checked={permissions.deviceControl}
                  onCheckedChange={(checked) => setPermissions({ ...permissions,
deviceControl: checked })}
                  className="data-[state=checked]:bg-[#00B2FF]"
                />
              </div>
            </CardContent>
          </Card>
        </motion.div>

        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ delay: 0.2 }}
          className="flex gap-4"
        >
          <Dialog open={isChangePinOpen} onOpenChange={setIsChangePinOpen}>
            <DialogTrigger asChild>
```

```
              <Button variant="outline" className="flex-1">
                <Key className="w-4 h-4 mr-2" />
                Change PIN
              </Button>
            </DialogTrigger>
            <DialogContent>
              <DialogHeader>
                <DialogTitle>Change PIN</DialogTitle>
                <DialogDescription>Enter a new 4-digit PIN for this profile.
</DialogDescription>
              </DialogHeader>
              <div className="flex justify-center gap-4 py-4">
                {[0, 1, 2, 3].map((i) => (
                  <Input
                    key={i}
                    id={`pin-${i}`}
                    type="text"
                    inputMode="numeric"
                    pattern="[0-9]*"
                    maxLength={1}
                    className="w-16 h-16 text-2xl text-center"
                    value={newPin[i]}
                    onChange={(e) => handlePinChange(i, e.target.value)}
                    onKeyDown={(e) => {
                      if (e.key === "Backspace" && !newPin[i] && i > 0) {
                        const prevInput = document.getElementById(`pin-${i - 1}`)
                        prevInput?.focus()
                      }
                    }}
                  />
                ))}
              </div>
              <DialogFooter>
                <Button
                  onClick={handleSavePin}
                  disabled={newPin.some((digit) => !digit)}
                  className="w-full bg-[#00B2FF] hover:bg-[#00B2FF]/90"
                >
                  Save PIN
                </Button>
              </DialogFooter>
            </DialogContent>
          </Dialog>

          <Button variant="destructive" className="flex-1" onClick=
{handleDeleteProfile}>
            <Trash2 className="w-4 h-4 mr-2" />
            Delete Profile
          </Button>

          <Button className="flex-1 bg-[#00B2FF] hover:bg-[#00B2FF]/90" onClick=
{handleSaveChanges}>
```

```
            Save Changes
          </Button>
        </motion.div>
      </div>
    </div>
  )
}
```

app/manage-profiles/page.tsx

```
// manage-profiles/page.tsx
"use client"

import { useState, useEffect } from "react"
import { Card, CardContent } from "@/components/ui/card"
import { Button } from "@/components/ui/button"
import { motion } from "framer-motion"
import { ArrowLeft, Users, Settings, Shield } from "lucide-react"
import { useRouter } from "next/navigation"
import Image from "next/image"

interface FamilyMember {
  id: string
  name: string
  email: string
  role: string
  avatar?: string
}

export default function ManageProfilesPage() {
  const router = useRouter()
  const [familyMembers, setFamilyMembers] = useState<FamilyMember[]>([])

  useEffect(() => {
    const storedMembers = JSON.parse(localStorage.getItem("familyMembers") || "[]")
    setFamilyMembers(storedMembers)
  }, [])

  return (
    <div className="p-6 min-h-screen bg-gradient-to-br from-gray-50 to-gray-100">
      <motion.div initial={{ opacity: 0, y: -20 }} animate={{ opacity: 1, y: 0 }}
transition={{ duration: 0.5 }}>
        <header className="flex justify-between items-center mb-8">
          <div className="flex items-center gap-4">
            <Button variant="ghost" size="icon" onClick={() => router.back()}>
              <ArrowLeft className="h-6 w-6" />
            </Button>
            <div className="flex items-center gap-4">
              <div className="w-12 h-12 bg-gradient-to-br from-[#00B2FF] to-
```

```
[#0085FF] rounded-full flex items-center justify-center shadow-lg">
                <Users className="h-6 w-6 text-white" />
              </div>
              <div>
                <h1 className="text-2xl font-bold text-gray-800">Manage
Profiles</h1>
                <p className="text-sm text-gray-500">Manage access and permissions
for family members</p>
              </div>
            </div>
          </div>
        </header>
      </motion.div>

      <motion.div
        initial={{ opacity: 0 }}
        animate={{ opacity: 1 }}
        transition={{ delay: 0.2 }}
        className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6"
      >
        {familyMembers.map((member, index) => (
          <motion.div
            key={member.id}
            initial={{ opacity: 0, y: 20 }}
            animate={{ opacity: 1, y: 0 }}
            transition={{ delay: index * 0.1 }}
          >
            <Card className="overflow-hidden hover:shadow-lg transition-shadow">
              <CardContent className="p-6">
                <div className="flex flex-col items-center text-center">
                  <div className="relative w-24 h-24 mb-4">
                    {member.avatar ? (
                      <Image
                        src={member.avatar || "/placeholder.svg"}
                        alt={member.name}
                        fill
                        className="rounded-full object-cover"
                      />
                    ) : (
                      <div className="w-full h-full bg-gray-100 rounded-full flex
items-center justify-center">
                        <span className="text-2xl font-semibold text-gray-500">
{member.name[0]}</span>
                      </div>
                    )}
                  </div>
                  <h3 className="font-semibold text-lg mb-1">{member.name}</h3>
                  <p className="text-sm text-gray-500 mb-2">
                    {member.role.charAt(0).toUpperCase() + member.role.slice(1)}
                  </p>
                  <p className="text-sm text-gray-500 mb-4">{member.email}</p>
                  <div className="grid grid-cols-2 gap-3 w-full">
```

```
                    <Button
                      variant="outline"
                      className="w-full text-[#00B2FF] border-[#00B2FF] hover:bg-
[#00B2FF]/10"
                      onClick={() => router.push(`/manage-
profiles/${member.id}/details`)}
                    >
                      <Settings className="w-4 h-4 mr-2" />
                      Manage Details
                    </Button>
                    <Button
                      className="w-full bg-[#00B2FF] hover:bg-[#00B2FF]/90"
                      onClick={() => router.push(`/manage-
profiles/${member.id}/access`)}
                    >
                      <Shield className="w-4 h-4 mr-2" />
                      Manage Access
                    </Button>
                  </div>
                </div>
              </CardContent>
            </Card>
          </motion.div>
        ))}
      </motion.div>
    </div>
  )
}
```

app/page.tsx

```tsx
// page.tsx
import { redirect } from "next/navigation"

export default function Page() {
  redirect("/dashboard")
}
```

app/profile/page.tsx

```tsx
// profile/page.tsx
"use client";

import type React from "react";

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

```tsx
import { Card, CardContent } from "@/components/ui/card";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Label } from "@/components/ui/label";
import {
  Dialog,
  DialogContent,
  DialogDescription,
  DialogFooter,
  DialogHeader,
  DialogTitle,
} from "@/components/ui/dialog";
import { useLogout } from "@/app/components/logout";
import { motion } from "framer-motion";
import { User, Settings, BarChart2, Users, LogOut, Plus, Lock } from "lucide-react";
import { useRouter } from "next/navigation";
import Link from "next/link";
import { toast } from "@/components/ui/use-toast";
import Image from "next/image";
import { NavigationSidebar } from "@/app/components/navigation-sidebar"; // Import
the navbar

interface FamilyMember {
  id: string;
  name: string;
  email: string;
  role: string;
}

export default function ProfilePage() {
  const [isEditProfileOpen, setIsEditProfileOpen] = useState(false);
  const [editProfileData, setEditProfileData] = useState({
    name: "",
    email: "",
  });
  const [user, setUser] = useState<any>(null);
  const logout = useLogout();
  const router = useRouter();

  useEffect(() => {
    const currentUser = JSON.parse(localStorage.getItem("currentUser") || "{}");
    if (!currentUser.email || !currentUser.isAuthenticated) {
      router.push("/auth/login");
      return;
    }
    setUser(currentUser);
    setEditProfileData({
      name: currentUser.name,
      email: currentUser.email,
    });
  }, [router]);
```

```
  const handleEditProfile = () => {
    // In a real application, this would update the user's information in the
backend
    setUser({ ...user, ...editProfileData });
    localStorage.setItem("currentUser", JSON.stringify({ ...user, ...editProfileData
}));
    setIsEditProfileOpen(false);
    toast({
      title: "Profile Updated",
      description: "Your profile information has been updated successfully.",
    });
  };

  const handleRequestAccess = (feature: string) => {
    // In a real application, this would send a request to the admin
    toast({
      title: "Access Requested",
      description: `Your request for access to ${feature} has been sent to the
admin.`,
    });
  };

  if (!user) return null;

  return (
    <div className="min-h-screen bg-gray-50 flex">
      <NavigationSidebar />
      <div className="flex-1 ml-[72px]">
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ duration: 0.5 }}
          className="p-6"
        >
          <motion.div initial={{ opacity: 0, y: 20 }} animate={{ opacity: 1, y: 0 }}
transition={{ duration: 0.5 }}>
            <Card className="max-w-4xl mx-auto">
              <CardContent className="p-6">
                <div className="flex justify-between items-start mb-8">
                  <div className="flex items-center gap-4">
                    <div className="w-16 h-16 bg-gradient-to-br from-[#00B2FF] to-
[#0085FF] rounded-full flex items-center justify-center shadow-lg">
                      <User className="w-8 h-8 text-white" />
                    </div>
                    <div>
                      <h1 className="text-2xl font-bold text-gray-800">Profile</h1>
                      <p className="text-sm text-gray-500">Manage your account and
family members</p>
                    </div>
                  </div>
                  <Button variant="outline" className="text-[#00B2FF]" onClick={()
=> setIsEditProfileOpen(true)}>
```

```
                    <Settings className="w-4 h-4 mr-2" />
                    Edit Profile
                  </Button>
                </div>

                <div className="flex flex-col md:flex-row items-center mb-8 bg-white
p-6 rounded-lg shadow-sm">
                  <div className="relative w-32 h-32 mb-4 md:mb-0 md:mr-6">
                    <Image src="/placeholder.svg" alt="Profile" layout="fill"
className="rounded-full object-cover" />
                  </div>
                  <div className="text-center md:text-left">
                    <h3 className="text-2xl font-semibold mb-1">{user.name}</h3>
                    <p className="text-gray-600 mb-1">{user.email}</p>
                    <p className="text-[#00B2FF] font-medium">{user.role}</p>
                    <div className="flex gap-2 mt-4">
                      <Button className="bg-[#00B2FF] hover:bg-[#00B2FF]/90"
onClick={() => setIsEditProfileOpen(true)}>
                        Edit Profile
                      </Button>
                      <Button variant="outline" onClick={logout}>
                        <LogOut className="w-4 h-4 mr-2" />
                        Logout
                      </Button>
                    </div>
                  </div>
                </div>

                <motion.div
                  initial={{ opacity: 0 }}
                  animate={{ opacity: 1 }}
                  transition={{ delay: 0.2 }}
                  className="grid grid-cols-1 md:grid-cols-2 gap-4"
                >
                  {user.type === "admin" ? (
                    <>
                      <Link href="/create-profile">
                        <Button className="w-full h-12 bg-[#00B2FF] text-white
hover:bg-[#00B2FF]/90">
                          <Plus className="w-4 h-4 mr-2" />
                          Create Profile
                        </Button>
                      </Link>

                      <Link href="/manage-profiles">
                        <Button
                          variant="outline"
                          className="w-full h-12 text-[#00B2FF] border-[#00B2FF]
hover:bg-[#00B2FF]/10"
                        >
                          <Users className="w-4 h-4 mr-2" />
                          Manage Profiles
```

```
                      </Button>
                    </Link>

                    <Button
                      variant="outline"
                      className="w-full h-12 text-[#00B2FF] border-[#00B2FF]
hover:bg-[#00B2FF]/10"
                      onClick={() => router.push("/statistics")}
                    >
                      <BarChart2 className="w-4 h-4 mr-2" />
                      View Analytics
                    </Button>

                    <Button
                      variant="outline"
                      className="w-full h-12 text-[#00B2FF] border-[#00B2FF]
hover:bg-[#00B2FF]/10"
                      onClick={() => router.push("/settings")}
                    >
                      <Settings className="w-4 h-4 mr-2" />
                      Account Settings
                    </Button>
                  </>
                ) : (
                  <>
                    {!user.permissions?.analyticalData && (
                      <Button
                        variant="outline"
                        className="w-full h-12 text-[#00B2FF] border-[#00B2FF]
hover:bg-[#00B2FF]/10"
                        onClick={() => handleRequestAccess("Analytical Data")}
                      >
                        <Lock className="w-4 h-4 mr-2" />
                        Request Analytics Access
                      </Button>
                    )}
                    {!user.permissions?.addAutomation && (
                      <Button
                        variant="outline"
                        className="w-full h-12 text-[#00B2FF] border-[#00B2FF]
hover:bg-[#00B2FF]/10"
                        onClick={() => handleRequestAccess("Automation")}
                      >
                        <Lock className="w-4 h-4 mr-2" />
                        Request Automation Access
                      </Button>
                    )}
                    <Button
                      variant="outline"
                      className="w-full h-12 text-[#00B2FF] border-[#00B2FF]
hover:bg-[#00B2FF]/10"
                      onClick={() => router.push("/settings")}
```

```
                    >
                      <Settings className="w-4 h-4 mr-2" />
                      Account Settings
                    </Button>
                  </>
                )}
              </motion.div>
            </CardContent>
          </Card>
        </motion.div>

        <Dialog open={isEditProfileOpen} onOpenChange={setIsEditProfileOpen}>
          <DialogContent className="sm:max-w-[425px]">
            <DialogHeader>
              <DialogTitle>Edit Profile</DialogTitle>
              <DialogDescription>Make changes to your profile here.
</DialogDescription>
            </DialogHeader>
            <div className="grid gap-4 py-4">
              <div className="grid grid-cols-4 items-center gap-4">
                <Label htmlFor="edit-name" className="text-right">
                  Name
                </Label>
                <Input
                  id="edit-name"
                  value={editProfileData.name}
                  onChange={(e) => setEditProfileData({ ...editProfileData, name:
e.target.value })}
                  className="col-span-3"
                />
              </div>
              <div className="grid grid-cols-4 items-center gap-4">
                <Label htmlFor="edit-email" className="text-right">
                  Email
                </Label>
                <Input
                  id="edit-email"
                  type="email"
                  value={editProfileData.email}
                  onChange={(e) => setEditProfileData({ ...editProfileData, email:
e.target.value })}
                  className="col-span-3"
                />
              </div>
            </div>
            <DialogFooter>
              <Button type="submit" onClick={handleEditProfile}>
                Save changes
              </Button>
            </DialogFooter>
          </DialogContent>
        </Dialog>
```

```
        </motion.div>
      </div>
    </div>
  );
}
```

app/rooms/add-room-dialog.tsx

```tsx
// rooms/add-room-dialog.tsx
"use client"

import type React from "react"

import { useState } from "react"
import { Dialog, DialogContent, DialogHeader, DialogTitle } from
"@/components/ui/dialog"
import { Input } from "@/components/ui/input"
import { Label } from "@/components/ui/label"
import { Button } from "@/components/ui/button"

interface AddRoomDialogProps {
  open: boolean
  onOpenChange: (open: boolean) => void
  onAdd: (room: { name: string; image: string }) => void
}

export function AddRoomDialog({ open, onOpenChange, onAdd }: AddRoomDialogProps) {
  const [name, setName] = useState("")
  const [image, setImage] = useState("")

  const handleSubmit = (e: React.FormEvent) => {
    e.preventDefault()
    if (name && image) {
      onAdd({ name, image })
      setName("")
      setImage("")
    }
  }

  return (
    <Dialog open={open} onOpenChange={onOpenChange}>
      <DialogContent>
        <DialogHeader>
          <DialogTitle>Add New Room</DialogTitle>
        </DialogHeader>
        <form onSubmit={handleSubmit} className="space-y-4">
          <div className="space-y-2">
            <Label htmlFor="name">Room Name</Label>
            <Input
              id="name"
```

```
              value={name}
              onChange={(e) => setName(e.target.value)}
              placeholder="Enter room name"
              required
            />
          </div>
          <div className="space-y-2">
            <Label htmlFor="image">Image URL</Label>
            <Input
              id="image"
              value={image}
              onChange={(e) => setImage(e.target.value)}
              placeholder="Enter image URL"
              required
            />
          </div>
          <Button type="submit" className="w-full">
            Add Room
          </Button>
        </form>
      </DialogContent>
    </Dialog>
  )
}
```

app/rooms/page.tsx

```
// rooms/page.tsx
"use client"

import { NavigationSidebar } from "../components/navigation-sidebar"
import { Button } from "@/components/ui/button"
import { Plus, User, Home } from "lucide-react"
import { useState } from "react"
import { RoomCard } from "./room-card"
import { AddRoomDialog } from "./add-room-dialog"
import { Card } from "@/components/ui/card"
import Link from "next/link"

const initialRooms = [
  {
    id: 1,
    name: "Living Room",
    image: "https://hebbkx1anhila5yf.public.blob.vercel-storage.com/IMG_0697-
MRkZkq9qJMNVm20BYSDpYOdkYHt3pF.jpeg",
  },
  {
    id: 2,
    name: "Bedroom",
```

```
    image: "https://hebbkx1anhila5yf.public.blob.vercel-storage.com/IMG_0697-
MRkZkq9qJMNVm20BYSDpYOdkYHt3pF.jpeg",
  },
  {
    id: 3,
    name: "Laundry Room",
    image: "https://hebbkx1anhila5yf.public.blob.vercel-storage.com/IMG_0697-
MRkZkq9qJMNVm20BYSDpYOdkYHt3pF.jpeg",
  },
  {
    id: 4,
    name: "Garage",
    image: "https://hebbkx1anhila5yf.public.blob.vercel-storage.com/IMG_0697-
MRkZkq9qJMNVm20BYSDpYOdkYHt3pF.jpeg",
  },
  {
    id: 5,
    name: "Kitchen",
    image: "https://hebbkx1anhila5yf.public.blob.vercel-storage.com/IMG_0697-
MRkZkq9qJMNVm20BYSDpYOdkYHt3pF.jpeg",
  },
]

export default function RoomsPage() {
  const [rooms, setRooms] = useState(initialRooms)
  const [isDialogOpen, setIsDialogOpen] = useState(false)

  const handleAddRoom = (newRoom: { name: string; image: string }) => {
    setRooms([...rooms, { id: rooms.length + 1, ...newRoom }])
    setIsDialogOpen(false)
  }

  return (
    <div className="min-h-screen bg-gray-50">
      <NavigationSidebar />
      <div className="ml-[72px] p-6">
        <header className="flex justify-between items-center mb-6">
          <div className="flex items-center gap-4">
            <div className="w-10 h-10 bg-[#00B2FF] rounded-full flex items-center
justify-center">
              <span className="text-white font-bold text-xl">Sy</span>
            </div>
            <h1 className="text-2xl font-bold">Room List</h1>
          </div>
          <Button variant="ghost" size="icon">
            <User className="h-5 w-5" />
          </Button>
        </header>

        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
          {rooms.length === 0 ? (
            <Card className="col-span-full p-6 text-center">
```

```
              <div className="flex flex-col items-center justify-center space-y-4">
                <div className="rounded-full bg-blue-100 p-3">
                  <Home className="h-8 w-8 text-[#00B2FF]" />
                </div>
                <div className="space-y-2">
                  <h3 className="font-semibold text-lg">No rooms added yet</h3>
                  <p className="text-muted-foreground">Start by adding your first
room to manage your smart home</p>
                </div>
                <Link href="/add-room">
                  <Button className="bg-[#00B2FF] hover:bg-[#00B2FF]/90">
                    <Plus className="mr-2 h-4 w-4" /> Add Your First Room
                  </Button>
                </Link>
              </div>
            </Card>
          ) : (
            <>
              {rooms.map((room) => (
                <RoomCard key={room.id} name={room.name} image={room.image} />
              ))}
              <button
                onClick={() => setIsDialogOpen(true)}
                className="aspect-video bg-white rounded-lg border-2 border-dashed
border-gray-200 flex flex-col items-center justify-center gap-2 hover:border-
[#00B2FF] transition-colors"
              >
                <Plus className="h-8 w-8 text-[#00B2FF]" />
                <span className="text-[#00B2FF] font-medium">Add a Room</span>
              </button>
            </>
          )}
        </div>

        <AddRoomDialog open={isDialogOpen} onOpenChange={setIsDialogOpen} onAdd=
{handleAddRoom} />
      </div>
    </div>
  )
}
```

app/rooms/room-card.tsx

```
// rooms/room-card.tsx
import Image from "next/image"
import { Card } from "@/components/ui/card"
import { Button } from "@/components/ui/button"
import { MoreVertical } from "lucide-react"
```

```
interface RoomCardProps {
  name: string
  image: string
}

export function RoomCard({ name, image }: RoomCardProps) {
  return (
    <Card className="overflow-hidden">
      <div className="relative aspect-video">
        <Image src={image || "/placeholder.svg"} alt={name} fill className="object-
cover" />
        <div className="absolute top-2 right-2">
          <Button variant="ghost" size="icon" className="bg-white/80 hover:bg-
white">
            <MoreVertical className="h-4 w-4" />
          </Button>
        </div>
        <div className="absolute bottom-2 left-2 bg-white/80 px-2 py-1 rounded text-
sm">{name}</div>
      </div>
    </Card>
  )
}
```

app/settings/page.tsx

```
// settings/page.tsx
"use client";

import { useState } from "react";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Label } from "@/components/ui/label";
import { Switch } from "@/components/ui/switch";
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from
"@/components/ui/select";
import { motion } from "framer-motion";
import {
  Bell,
  Moon,
  Sun,
  Shield,
  Wifi,
  Volume2,
  BellRing,
  Globe,
  HardDrive,
  RefreshCw,
```

```
  Smartphone,
  Zap,
  Settings2,
} from "lucide-react";
import { NavigationSidebar } from "@/app/components/navigation-sidebar"; // Import
the navbar

const container = {
  hidden: { opacity: 0 },
  show: {
    opacity: 1,
    transition: {
      staggerChildren: 0.1,
    },
  },
};

const item = {
  hidden: { opacity: 0, y: 20 },
  show: { opacity: 1, y: 0 },
};

export default function SettingsPage() {
  const [notifications, setNotifications] = useState(true);
  const [darkMode, setDarkMode] = useState(false);
  const [sound, setSound] = useState(true);

  return (
    <div className="min-h-screen bg-gradient-to-br from-gray-50 to-gray-100 flex">
      <NavigationSidebar />
      <div className="flex-1 ml-[72px]">
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ duration: 0.5 }}
          className="p-6"
        >
          <motion.div initial={{ opacity: 0, y: -20 }} animate={{ opacity: 1, y: 0
}} transition={{ duration: 0.5 }}>
            <header className="flex justify-between items-center mb-8">
              <div className="flex items-center gap-4">
                <div className="w-12 h-12 bg-gradient-to-br from-[#00B2FF] to-
[#0085FF] rounded-full flex items-center justify-center shadow-lg">
                  <Settings2 className="h-6 w-6 text-white" />
                </div>
                <div>
                  <h1 className="text-2xl font-bold text-gray-800">Settings</h1>
                  <p className="text-sm text-gray-500">Customize your smart home
experience</p>
                </div>
              </div>
            </header>
```

```
          </motion.div>

          <motion.div
            variants={container}
            initial="hidden"
            animate="show"
            className="grid grid-cols-1 md:grid-cols-2 gap-6"
          >
            <motion.div variants={item}>
              <Card className="overflow-hidden border-t-4 border-t-[#00B2FF]">
                <CardHeader>
                  <CardTitle className="flex items-center gap-2 text-[#00B2FF]">
                    <Zap className="h-5 w-5" />
                    General Settings
                  </CardTitle>
                </CardHeader>
                <CardContent className="space-y-6">
                  <div className="flex items-center justify-between p-4 bg-gray-50
rounded-lg transition-colors hover:bg-gray-100">
                    <div className="flex items-center gap-3">
                      <Bell className="h-5 w-5 text-[#00B2FF]" />
                      <div>
                        <p className="font-medium">Notifications</p>
                        <p className="text-sm text-gray-500">Enable push
notifications</p>
                      </div>
                    </div>
                    <Switch
                      checked={notifications}
                      onCheckedChange={setNotifications}
                      className="data-[state=checked]:bg-[#00B2FF]"
                    />
                  </div>

                  <div className="flex items-center justify-between p-4 bg-gray-50
rounded-lg transition-colors hover:bg-gray-100">
                    <div className="flex items-center gap-3">
                      <Moon className="h-5 w-5 text-[#00B2FF]" />
                      <div>
                        <p className="font-medium">Dark Mode</p>
                        <p className="text-sm text-gray-500">Toggle dark mode
theme</p>
                      </div>
                    </div>
                    <Switch
                      checked={darkMode}
                      onCheckedChange={setDarkMode}
                      className="data-[state=checked]:bg-[#00B2FF]"
                    />
                  </div>

                  <div className="flex items-center justify-between p-4 bg-gray-50
```

```
rounded-lg transition-colors hover:bg-gray-100">
                    <div className="flex items-center gap-3">
                      <Volume2 className="h-5 w-5 text-[#00B2FF]" />
                      <div>
                        <p className="font-medium">Sound Effects</p>
                        <p className="text-sm text-gray-500">Enable sound
effects</p>
                      </div>
                    </div>
                    <Switch checked={sound} onCheckedChange={setSound}
className="data-[state=checked]:bg-[#00B2FF]" />
                  </div>

                  <div className="space-y-2 p-4 bg-gray-50 rounded-lg transition-
colors hover:bg-gray-100">
                    <div className="flex items-center gap-3 mb-3">
                      <Globe className="h-5 w-5 text-[#00B2FF]" />
                      <Label className="font-medium">Language</Label>
                    </div>
                    <Select defaultValue="en">
                      <SelectTrigger className="w-full border-[#00B2FF] focus:ring-
[#00B2FF]">
                        <SelectValue />
                      </SelectTrigger>
                      <SelectContent>
                        <SelectItem value="en">English</SelectItem>
                        <SelectItem value="es">Spanish</SelectItem>
                        <SelectItem value="fr">French</SelectItem>
                        <SelectItem value="de">German</SelectItem>
                      </SelectContent>
                    </Select>
                  </div>
                </CardContent>
              </Card>
            </motion.div>

            <motion.div variants={item}>
              <Card className="overflow-hidden border-t-4 border-t-[#FF9500]">
                <CardHeader>
                  <CardTitle className="flex items-center gap-2 text-[#FF9500]">
                    <Shield className="h-5 w-5" />
                    Network & Security
                  </CardTitle>
                </CardHeader>
                <CardContent className="space-y-6">
                  <div className="space-y-2 p-4 bg-gray-50 rounded-lg transition-
colors hover:bg-gray-100">
                    <div className="flex items-center gap-3 mb-3">
                      <Wifi className="h-5 w-5 text-[#FF9500]" />
                      <Label className="font-medium">WiFi Network</Label>
                    </div>
                    <div className="flex gap-2">
```

```
                      <Input
                        value="MyHomeNetwork"
                        readOnly
                        className="flex-1 border-[#FF9500] focus-visible:ring-
[#FF9500]"
                      />
                      <Button
                        variant="outline"
                        size="icon"
                        className="text-[#FF9500] border-[#FF9500] hover:bg-
[#FF9500] hover:text-white"
                      >
                        <RefreshCw className="h-4 w-4" />
                      </Button>
                    </div>
                  </div>

                  <div className="space-y-2 p-4 bg-gray-50 rounded-lg transition-
colors hover:bg-gray-100">
                    <div className="flex items-center gap-3 mb-3">
                      <Shield className="h-5 w-5 text-[#FF9500]" />
                      <Label className="font-medium">Security Level</Label>
                    </div>
                    <Select defaultValue="high">
                      <SelectTrigger className="w-full border-[#FF9500] focus:ring-
[#FF9500]">
                        <SelectValue />
                      </SelectTrigger>
                      <SelectContent>
                        <SelectItem value="high">High Security</SelectItem>
                        <SelectItem value="medium">Medium Security</SelectItem>
                        <SelectItem value="low">Low Security</SelectItem>
                      </SelectContent>
                    </Select>
                  </div>

                  <div className="space-y-2 p-4 bg-gray-50 rounded-lg transition-
colors hover:bg-gray-100">
                    <div className="flex items-center gap-3 mb-3">
                      <HardDrive className="h-5 w-5 text-[#FF9500]" />
                      <Label className="font-medium">Backup Frequency</Label>
                    </div>
                    <Select defaultValue="daily">
                      <SelectTrigger className="w-full border-[#FF9500] focus:ring-
[#FF9500]">
                        <SelectValue />
                      </SelectTrigger>
                      <SelectContent>
                        <SelectItem value="daily">Daily</SelectItem>
                        <SelectItem value="weekly">Weekly</SelectItem>
                        <SelectItem value="monthly">Monthly</SelectItem>
                      </SelectContent>
```

```
                  </Select>
                </div>
              </CardContent>
            </Card>
          </motion.div>

          <motion.div variants={item}>
            <Card className="overflow-hidden border-t-4 border-t-[#00B2FF]">
              <CardHeader>
                <CardTitle className="flex items-center gap-2 text-[#00B2FF]">
                  <BellRing className="h-5 w-5" />
                  Notification Preferences
                </CardTitle>
              </CardHeader>
              <CardContent className="space-y-6">
                <div className="flex items-center justify-between p-4 bg-gray-50
rounded-lg transition-colors hover:bg-gray-100">
                  <div className="flex items-center gap-3">
                    <BellRing className="h-5 w-5 text-[#00B2FF]" />
                    <span className="font-medium">Device Alerts</span>
                  </div>
                  <Switch defaultChecked className="data-[state=checked]:bg-
[#00B2FF]" />
                </div>

                <div className="flex items-center justify-between p-4 bg-gray-50
rounded-lg transition-colors hover:bg-gray-100">
                  <div className="flex items-center gap-3">
                    <Shield className="h-5 w-5 text-[#00B2FF]" />
                    <span className="font-medium">Security Alerts</span>
                  </div>
                  <Switch defaultChecked className="data-[state=checked]:bg-
[#00B2FF]" />
                </div>

                <div className="flex items-center justify-between p-4 bg-gray-50
rounded-lg transition-colors hover:bg-gray-100">
                  <div className="flex items-center gap-3">
                    <Sun className="h-5 w-5 text-[#00B2FF]" />
                    <span className="font-medium">Energy Reports</span>
                  </div>
                  <Switch defaultChecked className="data-[state=checked]:bg-
[#00B2FF]" />
                </div>
              </CardContent>
            </Card>
          </motion.div>

          <motion.div variants={item}>
            <Card className="overflow-hidden border-t-4 border-t-[#FF9500]">
              <CardHeader>
                <CardTitle className="flex items-center gap-2 text-[#FF9500]">
```

```
                  <Smartphone className="h-5 w-5" />
                  System Information
                </CardTitle>
              </CardHeader>
              <CardContent className="space-y-4">
                <div className="p-4 bg-gray-50 rounded-lg transition-colors
hover:bg-gray-100">
                  <Label className="text-sm text-gray-500">System Version</Label>
                  <p className="font-medium text-[#FF9500]">2.1.0</p>
                </div>

                <div className="p-4 bg-gray-50 rounded-lg transition-colors
hover:bg-gray-100">
                  <Label className="text-sm text-gray-500">Last Updated</Label>
                  <p className="font-medium text-[#FF9500]">January 15, 2024</p>
                </div>

                <div className="p-4 bg-gray-50 rounded-lg transition-colors
hover:bg-gray-100">
                  <Label className="text-sm text-gray-500">Device ID</Label>
                  <p className="font-medium text-[#FF9500]">SYNC-HUB-001234</p>
                </div>

                <Button
                  variant="outline"
                  className="w-full h-12 mt-4 border-[#FF9500] text-[#FF9500]
hover:bg-[#FF9500] hover:text-white transition-colors"
                >
                  <RefreshCw className="w-4 h-4 mr-2" />
                  Check for Updates
                </Button>
              </CardContent>
            </Card>
          </motion.div>
        </motion.div>
      </motion.div>
    </div>
  </div>
  );
}
```

app/statistics/page.tsx

```
// statistics/page.tsx
"use client";

import { useState, useEffect } from "react";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";
import { Button } from "@/components/ui/button";
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";
```

```tsx
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from "@/components/ui/select";
import { LineChart, BarChart, PieChart } from "@/components/charts";
import { Download, User, BarChart2, PieChartIcon, TrendingUp, Lock } from "lucide-react";
import { PDFDocument, rgb } from "pdf-lib";
import { toPng } from "html-to-image";
//import { Inter } from "next/font/google";
import { motion } from "framer-motion";
import { useRouter } from "next/navigation";
import { useToast } from "@/components/ui/use-toast";
import { NavigationSidebar } from "@/app/components/navigation-sidebar"; // Import the navbar

//const inter = Inter({ subsets: ["latin"] });

interface Device {
  id: string;
  name: string;
  type: string;
  room: string;
  status: "on" | "off";
  powerConsumption: number;
  lastStatusChange: number;
  totalEnergyConsumed: number;
}

interface Room {
  id: string;
  name: string;
}

const generatePDF = async (devices: Device[], rooms: Room[]) => {
  const pdfDoc = await PDFDocument.create();
  const page = pdfDoc.addPage();
  const { width, height } = page.getSize();
  const fontSize = 12;

  // Add title
  page.drawText("Energy Consumption Report", {
    x: 50,
    y: height - 50,
    size: 20,
    color: rgb(0, 0, 0),
  });

  // Add date
  const currentDate = new Date().toLocaleDateString();
  page.drawText(`Generated on: ${currentDate}`, {
    x: 50,
    y: height - 80,
    size: fontSize,
```

```
    color: rgb(0.5, 0.5, 0.5),
  });

  // Add device usage table
  page.drawText("Device Usage", {
    x: 50,
    y: height - 120,
    size: 16,
    color: rgb(0, 0, 0),
  });

  devices.forEach((device, index) => {
    page.drawText(`${device.name} (${device.type}):
${device.totalEnergyConsumed.toFixed(2)} kWh`, {
      x: 50,
      y: height - 150 - index * 20,
      size: fontSize,
      color: rgb(0, 0, 0),
    });
  });

  // Add room usage table
  page.drawText("Room Usage", {
    x: 50,
    y: height - 300,
    size: 16,
    color: rgb(0, 0, 0),
  });

  rooms.forEach((room, index) => {
    const roomUsage = devices.filter((d) => d.room === room.name).reduce((sum, d) =>
sum + d.totalEnergyConsumed, 0);
    page.drawText(`${room.name}: ${roomUsage.toFixed(2)} kWh`, {
      x: 50,
      y: height - 330 - index * 20,
      size: fontSize,
      color: rgb(0, 0, 0),
    });
  });

  // Add charts
  const charts = document.querySelectorAll(".chart-container");
  for (let i = 0; i < charts.length; i++) {
    const chart = charts[i] as HTMLElement;

    // Add a style tag with the necessary font information
    const style = document.createElement("style");
    //style.textContent = `
    //  @import url('https://fonts.googleapis.com/css2?family=Inter&display=swap');
    //  * { font-family: 'Inter', sans-serif; }
    //`;
    chart.prepend(style);
```

```
    const pngImage = await toPng(chart, {
      quality: 0.95,
      backgroundColor: "#ffffff",
      style: {
        // Add any additional styles here if needed
      },
    });

    // Remove the added style tag
    chart.removeChild(style);

    const image = await pdfDoc.embedPng(pngImage);

    page.drawImage(image, {
      x: 50,
      y: height - 500 - i * 300,
      width: 500,
      height: 250,
    });
  }

  const pdfBytes = await pdfDoc.save();
  const blob = new Blob([pdfBytes], { type: "application/pdf" });
  const link = document.createElement("a");
  link.href = URL.createObjectURL(blob);
  link.download = "energy_statistics.pdf";
  link.click();
};

export default function StatisticsPage() {
  const router = useRouter();
  const [devices, setDevices] = useState<Device[]>([]);
  const [rooms, setRooms] = useState<Room[]>([]);
  const [timeRange, setTimeRange] = useState("day");
  const [user, setUser] = useState<any>(null);
  const { toast } = useToast();

  useEffect(() => {
    const currentUser = JSON.parse(localStorage.getItem("currentUser") || "{}");
    //if (!currentUser.id || !currentUser.permissions?.statisticalData) {
    //  router.push("/");
    //  return;
    //}
    setUser(currentUser);

    const storedDevices = JSON.parse(localStorage.getItem("devices") || "[]");
    const storedRooms = JSON.parse(localStorage.getItem("rooms") || "[]");
    setDevices(storedDevices);
    setRooms(storedRooms);
  }, [router]);
```

```
  const handleRequestAccess = () => {
    toast({
      title: "Access Requested",
      description: "Your request for access to Statistical Data has been sent to the
admin.",
    });
  };

  const filterDataByTimeRange = (data: any[]) => {
    if (data.length === 0) return [];

    const now = new Date();
    const startDate = new Date();

    switch (timeRange) {
      case "week":
        startDate.setDate(now.getDate() - 7);
        break;
      case "month":
        startDate.setMonth(now.getMonth() - 1);
        break;
      case "year":
        startDate.setFullYear(now.getFullYear() - 1);
        break;
      default: // day
        startDate.setDate(now.getDate() - 1);
    }

    return data.filter((item: any) => new Date(item.lastStatusChange) >= startDate);
  };

  const calculateRoomUsage = (roomName: string): number => {
    return devices
      .filter((device) => device.room === roomName)
      .reduce((total, device) => total + device.totalEnergyConsumed, 0);
  };

  const deviceData =
    devices.length > 0
      ? filterDataByTimeRange(devices).map((device) => ({
        name: device.name,
        usage: Number.parseFloat(device.totalEnergyConsumed.toFixed(2)),
      }))
      : [];

  const roomData = rooms
    .map((room) => ({
      name: room.name,
      usage: Number.parseFloat(calculateRoomUsage(room.name).toFixed(2)),
    }))
    .filter((room) => room.usage > 0);
```

```
  const deviceTypeData =
    devices.length > 0
      ? filterDataByTimeRange(devices).reduce(
        (acc, device) => {
          acc[device.type] = (acc[device.type] || 0) + device.totalEnergyConsumed;
          return acc;
        },
        {} as Record<string, number>
      )
      : {};

  const totalEnergyConsumed = devices.reduce((total, device) => total +
device.totalEnergyConsumed, 0);

  const containerVariants = {
    hidden: { opacity: 0 },
    visible: {
      opacity: 1,
      transition: {
        staggerChildren: 0.1,
      },
    },
  };

  const itemVariants = {
    hidden: { y: 20, opacity: 0 },
    visible: {
      y: 0,
      opacity: 1,
    },
  };

  if (!user) return null;

  if (user.type === "household" && !user.permissions?.statisticalData) {
    return (
      <div className="min-h-screen bg-gray-50 flex items-center justify-center">
        <Card className="w-full max-w-md">
          <CardContent className="flex flex-col items-center justify-center p-6">
            <Lock className="w-12 h-12 text-gray-400 mb-4" />
            <h3 className="text-lg font-semibold mb-2">Access Required</h3>
            <p className="text-sm text-gray-500 text-center mb-4">
              You don't have permission to view statistical data.
            </p>
            <Button onClick={handleRequestAccess}>Request Access</Button>
          </CardContent>
        </Card>
      </div>
    );
  }

  return (
```

```
    <div className="min-h-screen bg-gray-50 flex">
      <NavigationSidebar />
      <div className="flex-1 ml-[72px]">
        <motion.div
          initial="hidden"
          animate="visible"
          variants={containerVariants}
          className="p-6"
        >
          <motion.header variants={itemVariants} className="flex justify-between
items-center mb-8">
            <div className="flex items-center gap-4">
              <div className="w-12 h-12 bg-gradient-to-br from-[#00B2FF] to-
[#0085FF] rounded-full flex items-center justify-center shadow-lg">
                <BarChart2 className="h-6 w-6 text-white" />
              </div>
              <div>
                <h1 className="text-2xl font-bold text-gray-800">Energy
Statistics</h1>
                <p className="text-sm text-gray-500">Monitor and analyze your energy
consumption</p>
              </div>
            </div>
            <div className="flex items-center gap-4">
              <Select value={timeRange} onValueChange={setTimeRange}>
                <SelectTrigger className="w-[180px] border-[#00B2FF] text-
[#00B2FF]">
                  <SelectValue placeholder="Select time range" />
                </SelectTrigger>
                <SelectContent>
                  <SelectItem value="day">Past 24 Hours</SelectItem>
                  <SelectItem value="week">Past Week</SelectItem>
                  <SelectItem value="month">Past Month</SelectItem>
                  <SelectItem value="year">Past Year</SelectItem>
                </SelectContent>
              </Select>
              <Button
                variant="outline"
                size="icon"
                onClick={() => generatePDF(devices, rooms)}
                className="text-[#FF9500] border-[#FF9500] hover:bg-[#FF9500]
hover:text-white"
              >
                <Download className="h-5 w-5" />
              </Button>
              <Button
                variant="outline"
                size="icon"
                className="text-[#00B2FF] border-[#00B2FF] hover:bg-[#00B2FF]
hover:text-white"
              >
                <User className="h-5 w-5" />
```

```
              </Button>
            </div>
          </motion.header>

          <motion.div variants={containerVariants} className="grid grid-cols-1
md:grid-cols-3 gap-6 mb-8">
            <motion.div variants={itemVariants}>
              <Card className="bg-gradient-to-br from-[#00B2FF] to-[#0085FF] text-
white">
                <CardHeader>
                  <CardTitle className="text-lg font-medium">Total Energy
Consumed</CardTitle>
                </CardHeader>
                <CardContent>
                  <div className="text-3xl font-bold">
{totalEnergyConsumed.toFixed(2)} kWh</div>
                  <p className="text-sm opacity-80">in the selected time range</p>
                </CardContent>
              </Card>
            </motion.div>
            <motion.div variants={itemVariants}>
              <Card className="bg-gradient-to-br from-[#FF9500] to-[#FFB800] text-
white">
                <CardHeader>
                  <CardTitle className="text-lg font-medium">Active
Devices</CardTitle>
                </CardHeader>
                <CardContent>
                  <div className="text-3xl font-bold">{devices.filter((d) =>
d.status === "on").length}</div>
                  <p className="text-sm opacity-80">out of {devices.length} total
devices</p>
                </CardContent>
              </Card>
            </motion.div>
            <motion.div variants={itemVariants}>
              <Card className="bg-gradient-to-br from-[#00B2FF] to-[#0085FF] text-
white">
                <CardHeader>
                  <CardTitle className="text-lg font-medium">Most Active
Room</CardTitle>
                </CardHeader>
                <CardContent>
                  <div className="text-3xl font-bold">
                    {roomData.length > 0 ? roomData.reduce((a, b) => (a.usage >
b.usage ? a : b)).name : "No data"}
                  </div>
                  <p className="text-sm opacity-80">based on energy consumption</p>
                </CardContent>
              </Card>
            </motion.div>
          </motion.div>
```

```
          <motion.div variants={itemVariants}>
            <Tabs defaultValue="devices" className="space-y-6">
              <TabsList className="bg-white shadow-md rounded-lg p-1">
                <TabsTrigger value="devices" className="data-[state=active]:bg-
[#00B2FF] data-[state=active]:text-white">
                  <BarChart2 className="h-4 w-4 mr-2" />
                  Devices
                </TabsTrigger>
                <TabsTrigger value="rooms" className="data-[state=active]:bg-
[#00B2FF] data-[state=active]:text-white">
                  <PieChartIcon className="h-4 w-4 mr-2" />
                  Rooms
                </TabsTrigger>
                <TabsTrigger value="overview" className="data-[state=active]:bg-
[#00B2FF] data-[state=active]:text-white">
                  <TrendingUp className="h-4 w-4 mr-2" />
                  Overview
                </TabsTrigger>
              </TabsList>
              <TabsContent value="devices" className="space-y-6">
                <Card>
                  <CardHeader>
                    <CardTitle className="text-xl font-semibold text-gray-
800">Energy Usage by Device</CardTitle>
                  </CardHeader>
                  <CardContent>
                    {deviceData.length > 0 ? (
                      <BarChart data={deviceData} />
                    ) : (
                      <p className="text-center text-gray-500">No data available for
the selected time range</p>
                    )}
                  </CardContent>
                </Card>
                <Card>
                  <CardHeader>
                    <CardTitle className="text-xl font-semibold text-gray-
800">Device Usage Over Time</CardTitle>
                  </CardHeader>
                  <CardContent>
                    {deviceData.length > 0 ? (
                      <LineChart data={deviceData} />
                    ) : (
                      <p className="text-center text-gray-500">No data available for
the selected time range</p>
                    )}
                  </CardContent>
                </Card>
              </TabsContent>
              <TabsContent value="rooms" className="space-y-6">
                <Card>
```

```
                <CardHeader>
                  <CardTitle className="text-xl font-semibold text-gray-
800">Energy Usage by Room</CardTitle>
                </CardHeader>
                <CardContent>
                  {roomData.length > 0 ? (
                    <BarChart data={roomData} />
                  ) : (
                    <p className="text-center text-gray-500">No data available for
the selected time range</p>
                  )}
                </CardContent>
              </Card>
              <Card>
                <CardHeader>
                  <CardTitle className="text-xl font-semibold text-gray-800">Room
Usage Over Time</CardTitle>
                </CardHeader>
                <CardContent>
                  {roomData.length > 0 ? (
                    <LineChart data={roomData} />
                  ) : (
                    <p className="text-center text-gray-500">No data available for
the selected time range</p>
                  )}
                </CardContent>
              </Card>
            </TabsContent>
            <TabsContent value="overview" className="space-y-6">
              <Card>
                <CardHeader>
                  <CardTitle className="text-xl font-semibold text-gray-
800">Energy Usage by Device Type</CardTitle>
                </CardHeader>
                <CardContent>
                  {Object.keys(deviceTypeData).length > 0 ? (
                    <PieChart
                      data={Object.entries(deviceTypeData).map(([name, usage]) =>
({
                        name,
                        usage: Number.parseFloat(usage.toFixed(2)),
                      }))}
                    />
                  ) : (
                    <p className="text-center text-gray-500">No data available for
the selected time range</p>
                  )}
                </CardContent>
              </Card>
              <Card>
                <CardHeader>
                  <CardTitle className="text-xl font-semibold text-gray-800">Total
```

```
Energy Consumption Trend</CardTitle>
                  </CardHeader>
                  <CardContent>
                    {deviceData.length > 0 ? (
                      <LineChart data={deviceData} />
                    ) : (
                      <p className="text-center text-gray-500">No data available for
the selected time range</p>
                    )}
                  </CardContent>
                </Card>
              </TabsContent>
            </Tabs>
          </motion.div>
        </motion.div>
      </div>
    </div>
  );
}
```

app/statistics/page.tsx.bak

```
"use client";

import { useState, useEffect } from "react";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";
import { Button } from "@/components/ui/button";
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from
"@/components/ui/select";
import { LineChart, BarChart, PieChart } from "@/components/charts";
import { Download, User, BarChart2, PieChartIcon, TrendingUp, Lock } from "lucide-
react";
import { PDFDocument, rgb } from "pdf-lib";
import { toPng } from "html-to-image";
import { Inter } from "next/font/google";
import { motion } from "framer-motion";
import { useRouter } from "next/navigation";
import { useToast } from "@/components/ui/use-toast";
import { NavigationSidebar } from "@/app/components/navigation-sidebar"; // Import
the navbar

const inter = Inter({ subsets: ["latin"] });

interface Device {
  id: string;
  name: string;
  type: string;
  room: string;
  status: "on" | "off";
```

```
  powerConsumption: number;
  lastStatusChange: number;
  totalEnergyConsumed: number;
}

interface Room {
  id: string;
  name: string;
}

const generatePDF = async (devices: Device[], rooms: Room[]) => {
  const pdfDoc = await PDFDocument.create();
  const page = pdfDoc.addPage();
  const { width, height } = page.getSize();
  const fontSize = 12;

  // Add title
  page.drawText("Energy Consumption Report", {
    x: 50,
    y: height - 50,
    size: 20,
    color: rgb(0, 0, 0),
  });

  // Add date
  const currentDate = new Date().toLocaleDateString();
  page.drawText(`Generated on: ${currentDate}`, {
    x: 50,
    y: height - 80,
    size: fontSize,
    color: rgb(0.5, 0.5, 0.5),
  });

  // Add device usage table
  page.drawText("Device Usage", {
    x: 50,
    y: height - 120,
    size: 16,
    color: rgb(0, 0, 0),
  });

  devices.forEach((device, index) => {
    page.drawText(`${device.name} (${device.type}):
${device.totalEnergyConsumed.toFixed(2)} kWh`, {
      x: 50,
      y: height - 150 - index * 20,
      size: fontSize,
      color: rgb(0, 0, 0),
    });
  });

  // Add room usage table
```

```
  page.drawText("Room Usage", {
    x: 50,
    y: height - 300,
    size: 16,
    color: rgb(0, 0, 0),
  });

  rooms.forEach((room, index) => {
    const roomUsage = devices.filter((d) => d.room === room.name).reduce((sum, d) =>
sum + d.totalEnergyConsumed, 0);
    page.drawText(`${room.name}: ${roomUsage.toFixed(2)} kWh`, {
      x: 50,
      y: height - 330 - index * 20,
      size: fontSize,
      color: rgb(0, 0, 0),
    });
  });

  // Add charts
  const charts = document.querySelectorAll(".chart-container");
  for (let i = 0; i < charts.length; i++) {
    const chart = charts[i] as HTMLElement;

    // Add a style tag with the necessary font information
    const style = document.createElement("style");
    style.textContent = `
      @import url('https://fonts.googleapis.com/css2?family=Inter&display=swap');
      * { font-family: 'Inter', sans-serif; }
    `;
    chart.prepend(style);

    const pngImage = await toPng(chart, {
      quality: 0.95,
      backgroundColor: "#ffffff",
      style: {
        // Add any additional styles here if needed
      },
    });

    // Remove the added style tag
    chart.removeChild(style);

    const image = await pdfDoc.embedPng(pngImage);

    page.drawImage(image, {
      x: 50,
      y: height - 500 - i * 300,
      width: 500,
      height: 250,
    });
  }
```

```
  const pdfBytes = await pdfDoc.save();
  const blob = new Blob([pdfBytes], { type: "application/pdf" });
  const link = document.createElement("a");
  link.href = URL.createObjectURL(blob);
  link.download = "energy_statistics.pdf";
  link.click();
};

export default function StatisticsPage() {
  const router = useRouter();
  const [devices, setDevices] = useState<Device[]>([]);
  const [rooms, setRooms] = useState<Room[]>([]);
  const [timeRange, setTimeRange] = useState("day");
  const [user, setUser] = useState<any>(null);
  const { toast } = useToast();

  useEffect(() => {
    const currentUser = JSON.parse(localStorage.getItem("currentUser") || "{}");
    if (!currentUser.id || !currentUser.permissions?.statisticalData) {
      router.push("/");
      return;
    }
    setUser(currentUser);

    const storedDevices = JSON.parse(localStorage.getItem("devices") || "[]");
    const storedRooms = JSON.parse(localStorage.getItem("rooms") || "[]");
    setDevices(storedDevices);
    setRooms(storedRooms);
  }, [router]);

  const handleRequestAccess = () => {
    toast({
      title: "Access Requested",
      description: "Your request for access to Statistical Data has been sent to the
admin.",
    });
  };

  const filterDataByTimeRange = (data: any[]) => {
    if (data.length === 0) return [];

    const now = new Date();
    const startDate = new Date();

    switch (timeRange) {
      case "week":
        startDate.setDate(now.getDate() - 7);
        break;
      case "month":
        startDate.setMonth(now.getMonth() - 1);
        break;
      case "year":
```

```
        startDate.setFullYear(now.getFullYear() - 1);
        break;
      default: // day
        startDate.setDate(now.getDate() - 1);
    }

    return data.filter((item: any) => new Date(item.lastStatusChange) >= startDate);
  };

  const calculateRoomUsage = (roomName: string): number => {
    return devices
      .filter((device) => device.room === roomName)
      .reduce((total, device) => total + device.totalEnergyConsumed, 0);
  };

  const deviceData =
    devices.length > 0
      ? filterDataByTimeRange(devices).map((device) => ({
        name: device.name,
        usage: Number.parseFloat(device.totalEnergyConsumed.toFixed(2)),
      }))
      : [];

  const roomData = rooms
    .map((room) => ({
      name: room.name,
      usage: Number.parseFloat(calculateRoomUsage(room.name).toFixed(2)),
    }))
    .filter((room) => room.usage > 0);

  const deviceTypeData =
    devices.length > 0
      ? filterDataByTimeRange(devices).reduce(
        (acc, device) => {
          acc[device.type] = (acc[device.type] || 0) + device.totalEnergyConsumed;
          return acc;
        },
        {} as Record<string, number>
      )
      : {};

  const totalEnergyConsumed = devices.reduce((total, device) => total +
device.totalEnergyConsumed, 0);

  const containerVariants = {
    hidden: { opacity: 0 },
    visible: {
      opacity: 1,
      transition: {
        staggerChildren: 0.1,
      },
    },
```

```
  };

  const itemVariants = {
    hidden: { y: 20, opacity: 0 },
    visible: {
      y: 0,
      opacity: 1,
    },
  };

  if (!user) return null;

  if (user.type === "household" && !user.permissions?.statisticalData) {
    return (
      <div className="min-h-screen bg-gray-50 flex items-center justify-center">
        <Card className="w-full max-w-md">
          <CardContent className="flex flex-col items-center justify-center p-6">
            <Lock className="w-12 h-12 text-gray-400 mb-4" />
            <h3 className="text-lg font-semibold mb-2">Access Required</h3>
            <p className="text-sm text-gray-500 text-center mb-4">
              You don't have permission to view statistical data.
            </p>
            <Button onClick={handleRequestAccess}>Request Access</Button>
          </CardContent>
        </Card>
      </div>
    );
  }

  return (
    <div className="min-h-screen bg-gray-50 flex">
      <NavigationSidebar />
      <div className="flex-1 ml-[72px]">
        <motion.div
          initial="hidden"
          animate="visible"
          variants={containerVariants}
          className="p-6"
        >
          <motion.header variants={itemVariants} className="flex justify-between
items-center mb-8">
            <div className="flex items-center gap-4">
              <div className="w-12 h-12 bg-gradient-to-br from-[#00B2FF] to-
[#0085FF] rounded-full flex items-center justify-center shadow-lg">
                <BarChart2 className="h-6 w-6 text-white" />
              </div>
              <div>
                <h1 className="text-2xl font-bold text-gray-800">Energy
Statistics</h1>
                <p className="text-sm text-gray-500">Monitor and analyze your energy
consumption</p>
              </div>
```

```
            </div>
            <div className="flex items-center gap-4">
              <Select value={timeRange} onValueChange={setTimeRange}>
                <SelectTrigger className="w-[180px] border-[#00B2FF] text-
[#00B2FF]">
                  <SelectValue placeholder="Select time range" />
                </SelectTrigger>
                <SelectContent>
                  <SelectItem value="day">Past 24 Hours</SelectItem>
                  <SelectItem value="week">Past Week</SelectItem>
                  <SelectItem value="month">Past Month</SelectItem>
                  <SelectItem value="year">Past Year</SelectItem>
                </SelectContent>
              </Select>
              <Button
                variant="outline"
                size="icon"
                onClick={() => generatePDF(devices, rooms)}
                className="text-[#FF9500] border-[#FF9500] hover:bg-[#FF9500]
hover:text-white"
              >
                <Download className="h-5 w-5" />
              </Button>
              <Button
                variant="outline"
                size="icon"
                className="text-[#00B2FF] border-[#00B2FF] hover:bg-[#00B2FF]
hover:text-white"
              >
                <User className="h-5 w-5" />
              </Button>
            </div>
          </motion.header>

          <motion.div variants={containerVariants} className="grid grid-cols-1
md:grid-cols-3 gap-6 mb-8">
            <motion.div variants={itemVariants}>
              <Card className="bg-gradient-to-br from-[#00B2FF] to-[#0085FF] text-
white">
                <CardHeader>
                  <CardTitle className="text-lg font-medium">Total Energy
Consumed</CardTitle>
                </CardHeader>
                <CardContent>
                  <div className="text-3xl font-bold">
{totalEnergyConsumed.toFixed(2)} kWh</div>
                  <p className="text-sm opacity-80">in the selected time range</p>
                </CardContent>
              </Card>
            </motion.div>
            <motion.div variants={itemVariants}>
              <Card className="bg-gradient-to-br from-[#FF9500] to-[#FFB800] text-
```

```
white">
                <CardHeader>
                  <CardTitle className="text-lg font-medium">Active
Devices</CardTitle>
                </CardHeader>
                <CardContent>
                  <div className="text-3xl font-bold">{devices.filter((d) =>
d.status === "on").length}</div>
                  <p className="text-sm opacity-80">out of {devices.length} total
devices</p>
                </CardContent>
              </Card>
            </motion.div>
            <motion.div variants={itemVariants}>
              <Card className="bg-gradient-to-br from-[#00B2FF] to-[#0085FF] text-
white">
                <CardHeader>
                  <CardTitle className="text-lg font-medium">Most Active
Room</CardTitle>
                </CardHeader>
                <CardContent>
                  <div className="text-3xl font-bold">
                    {roomData.length > 0 ? roomData.reduce((a, b) => (a.usage >
b.usage ? a : b)).name : "No data"}
                  </div>
                  <p className="text-sm opacity-80">based on energy consumption</p>
                </CardContent>
              </Card>
            </motion.div>
          </motion.div>

          <motion.div variants={itemVariants}>
            <Tabs defaultValue="devices" className="space-y-6">
              <TabsList className="bg-white shadow-md rounded-lg p-1">
                <TabsTrigger value="devices" className="data-[state=active]:bg-
[#00B2FF] data-[state=active]:text-white">
                  <BarChart2 className="h-4 w-4 mr-2" />
                  Devices
                </TabsTrigger>
                <TabsTrigger value="rooms" className="data-[state=active]:bg-
[#00B2FF] data-[state=active]:text-white">
                  <PieChartIcon className="h-4 w-4 mr-2" />
                  Rooms
                </TabsTrigger>
                <TabsTrigger value="overview" className="data-[state=active]:bg-
[#00B2FF] data-[state=active]:text-white">
                  <TrendingUp className="h-4 w-4 mr-2" />
                  Overview
                </TabsTrigger>
              </TabsList>
              <TabsContent value="devices" className="space-y-6">
                <Card>
```

```
                <CardHeader>
                  <CardTitle className="text-xl font-semibold text-gray-
800">Energy Usage by Device</CardTitle>
                </CardHeader>
                <CardContent>
                  {deviceData.length > 0 ? (
                    <BarChart data={deviceData} />
                  ) : (
                    <p className="text-center text-gray-500">No data available for
the selected time range</p>
                  )}
                </CardContent>
              </Card>
              <Card>
                <CardHeader>
                  <CardTitle className="text-xl font-semibold text-gray-
800">Device Usage Over Time</CardTitle>
                </CardHeader>
                <CardContent>
                  {deviceData.length > 0 ? (
                    <LineChart data={deviceData} />
                  ) : (
                    <p className="text-center text-gray-500">No data available for
the selected time range</p>
                  )}
                </CardContent>
              </Card>
            </TabsContent>
            <TabsContent value="rooms" className="space-y-6">
              <Card>
                <CardHeader>
                  <CardTitle className="text-xl font-semibold text-gray-
800">Energy Usage by Room</CardTitle>
                </CardHeader>
                <CardContent>
                  {roomData.length > 0 ? (
                    <BarChart data={roomData} />
                  ) : (
                    <p className="text-center text-gray-500">No data available for
the selected time range</p>
                  )}
                </CardContent>
              </Card>
              <Card>
                <CardHeader>
                  <CardTitle className="text-xl font-semibold text-gray-800">Room
Usage Over Time</CardTitle>
                </CardHeader>
                <CardContent>
                  {roomData.length > 0 ? (
                    <LineChart data={roomData} />
                  ) : (
```

```
                    <p className="text-center text-gray-500">No data available for
the selected time range</p>
                  )}
                </CardContent>
              </Card>
            </TabsContent>
            <TabsContent value="overview" className="space-y-6">
              <Card>
                <CardHeader>
                  <CardTitle className="text-xl font-semibold text-gray-
800">Energy Usage by Device Type</CardTitle>
                </CardHeader>
                <CardContent>
                  {Object.keys(deviceTypeData).length > 0 ? (
                    <PieChart
                      data={Object.entries(deviceTypeData).map(([name, usage]) =>
({
                        name,
                        usage: Number.parseFloat(usage.toFixed(2)),
                      }))}
                    />
                  ) : (
                    <p className="text-center text-gray-500">No data available for
the selected time range</p>
                  )}
                </CardContent>
              </Card>
              <Card>
                <CardHeader>
                  <CardTitle className="text-xl font-semibold text-gray-800">Total
Energy Consumption Trend</CardTitle>
                </CardHeader>
                <CardContent>
                  {deviceData.length > 0 ? (
                    <LineChart data={deviceData} />
                  ) : (
                    <p className="text-center text-gray-500">No data available for
the selected time range</p>
                  )}
                </CardContent>
              </Card>
            </TabsContent>
          </Tabs>
        </motion.div>
      </motion.div>
    </div>
  </div>
  );
}
```

app/statistics/page.tsx.bak.bak

```
"use client";

import { useState, useEffect } from "react";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card";
import { Button } from "@/components/ui/button";
import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from
"@/components/ui/select";
import { LineChart, BarChart, PieChart } from "@/components/charts";
import { Download, User, BarChart2, PieChartIcon, TrendingUp, Lock, FileText } from
"lucide-react";
import { PDFDocument, rgb } from "pdf-lib";
import { toPng } from "html-to-image";
import { motion } from "framer-motion";
import { useRouter } from "next/navigation";
import { useToast } from "@/components/ui/use-toast";
import { NavigationSidebar } from "@/app/components/navigation-sidebar"; // Import
the navbar
import {
  Dialog,
  DialogContent,
  DialogDescription,
  DialogHeader,
  DialogTitle,
  DialogTrigger,
  DialogFooter,
  DialogClose,
} from "@/components/ui/dialog";
import generateEnhancedReport from "@/components/generate-enhanced-reports";

interface Device {
  id: string;
  name: string;
  type: string;
  room: string;
  status: "on" | "off";
  powerConsumption: number;
  lastStatusChange: number;
  totalEnergyConsumed: number;
}

interface Room {
  id: string;
  name: string;
}

// Original PDF generation function (kept for backward compatibility)
const generatePDF = async (devices: Device[], rooms: Room[]) => {
```

```
  const pdfDoc = await PDFDocument.create();
  const page = pdfDoc.addPage();
  const { width, height } = page.getSize();
  const fontSize = 12;

  // Add title
  page.drawText("Energy Consumption Report", {
    x: 50,
    y: height - 50,
    size: 20,
    color: rgb(0, 0, 0),
  });

  // Add date
  const currentDate = new Date().toLocaleDateString();
  page.drawText(`Generated on: ${currentDate}`, {
    x: 50,
    y: height - 80,
    size: fontSize,
    color: rgb(0.5, 0.5, 0.5),
  });

  // Add device usage table
  page.drawText("Device Usage", {
    x: 50,
    y: height - 120,
    size: 16,
    color: rgb(0, 0, 0),
  });

  devices.forEach((device, index) => {
    page.drawText(`${device.name} (${device.type}):
${device.totalEnergyConsumed.toFixed(2)} kWh`, {
      x: 50,
      y: height - 150 - index * 20,
      size: fontSize,
      color: rgb(0, 0, 0),
    });
  });

  // Add room usage table
  page.drawText("Room Usage", {
    x: 50,
    y: height - 300,
    size: 16,
    color: rgb(0, 0, 0),
  });

  rooms.forEach((room, index) => {
    const roomUsage = devices.filter((d) => d.room === room.name).reduce((sum, d) =>
sum + d.totalEnergyConsumed, 0);
    page.drawText(`${room.name}: ${roomUsage.toFixed(2)} kWh`, {
```

```
      x: 50,
      y: height - 330 - index * 20,
      size: fontSize,
      color: rgb(0, 0, 0),
    });
  });

  // Add charts
  const charts = document.querySelectorAll(".chart-container");
  for (let i = 0; i < charts.length; i++) {
    const chart = charts[i] as HTMLElement;

    // Add a style tag with the necessary font information
    const style = document.createElement("style");
    chart.prepend(style);

    const pngImage = await toPng(chart, {
      quality: 0.95,
      backgroundColor: "#ffffff",
      style: {
        // Add any additional styles here if needed
      },
    });

    // Remove the added style tag
    chart.removeChild(style);

    const image = await pdfDoc.embedPng(pngImage);

    page.drawImage(image, {
      x: 50,
      y: height - 500 - i * 300,
      width: 500,
      height: 250,
    });
  }

  const pdfBytes = await pdfDoc.save();
  const blob = new Blob([pdfBytes], { type: "application/pdf" });
  const link = document.createElement("a");
  link.href = URL.createObjectURL(blob);
  link.download = "energy_statistics.pdf";
  link.click();
};

// New function to download enhanced reports from the backend
const downloadEnhancedReport = async (format: string, timeRange: string, toast: any)
=> {
  try {
    // Convert time range to date range
    const endDate = new Date().toISOString().split('T')[0]; // Today in YYYY-MM-DD
format
```

```
    let startDate;

    switch (timeRange) {
      case 'week':
        startDate = new Date(Date.now() - 7 * 24 * 60 * 60 *
1000).toISOString().split('T')[0];
      break;
      case 'month':
        startDate = new Date(Date.now() - 30 * 24 * 60 * 60 *
1000).toISOString().split('T')[0];
      break;
      case 'year':
        startDate = new Date(Date.now() - 365 * 24 * 60 * 60 *
1000).toISOString().split('T')[0];
      break;
      default: // day
        startDate = new Date(Date.now() - 1 * 24 * 60 * 60 *
1000).toISOString().split('T')[0];
    }

    // Get auth token from local storage, handling potential parsing errors
    let token = "";
    try {
      // Try to get the user object
      const userStr = localStorage.getItem("currentUser");
      if (userStr) {
        const user = JSON.parse(userStr);
        token = user.token || localStorage.getItem("authToken") || "";
      } else {
        // If no user object, try a direct token
        token = localStorage.getItem("authToken") || "";
      }
    } catch (error) {
      console.error("Error parsing user data from localStorage:", error);
      // Try alternate token storage if the user object can't be parsed
      token = localStorage.getItem("authToken") || "";
    }

    if (!token) {
      toast({
        title: "Authentication Error",
        description: "You must be logged in to download reports.",
        variant: "destructive",
      });
      return;
    }

    // Call backend API
    // Check if the token already has "Bearer " prefix, if not add it
    const authToken = token.startsWith('Bearer ') ? token : `Bearer ${token}`;

    console.log(`Making request to backend with time range: ${timeRange}
```

```
(${startDate} to ${endDate})`);

    const response = await fetch(`/api/v1/reports/report?
format=${format}&start_date=${startDate}&end_date=${endDate}`, {
      method: 'POST',
      headers: {
        'Authorization': authToken,
        'Accept': 'application/octet-stream',
      },
    });

    if (!response.ok) {
      const errorData = await response.json();
      throw new Error(errorData.detail || "Failed to download report");
    }

    // Handle the file download
    const blob = await response.blob();
    const fileExtension = format === 'pdf' ? 'pdf' : 'csv';
    const fileType = format === 'pdf' ? 'application/pdf' : 'text/csv';

    // Create a download link and trigger it
    const fileURL = URL.createObjectURL(new Blob([blob], { type: fileType }));
    const downloadLink = document.createElement('a');
    downloadLink.href = fileURL;
    downloadLink.download = `energy_report_${new Date().toISOString().split('T')
[0]}.${fileExtension}`;
    document.body.appendChild(downloadLink);
    downloadLink.click();
    document.body.removeChild(downloadLink);

    toast({
      title: "Report Downloaded",
      description: `Your ${format.toUpperCase()} report has been downloaded
successfully.`,
      variant: "default",
    });

  } catch (error) {
    console.error("Error downloading report:", error);
    toast({
      title: "Download Failed",
      description: error instanceof Error ? error.message : "Failed to download
report. Please try again.",
      variant: "destructive",
    });
  }
};

export default function StatisticsPage() {
  const router = useRouter();
  const [devices, setDevices] = useState<Device[]>([]);
```

```
  const [rooms, setRooms] = useState<Room[]>([]);
  const [timeRange, setTimeRange] = useState("day");
  const [user, setUser] = useState<any>(null);
  const [isReportDialogOpen, setIsReportDialogOpen] = useState(false);
  const { toast } = useToast();

  useEffect(() => {
    const currentUser = JSON.parse(localStorage.getItem("currentUser") || "{}");
    setUser(currentUser);

    const storedDevices = JSON.parse(localStorage.getItem("devices") || "[]");
    const storedRooms = JSON.parse(localStorage.getItem("rooms") || "[]");
    setDevices(storedDevices);
    setRooms(storedRooms);
  }, [router]);

  const handleRequestAccess = () => {
    toast({
      title: "Access Requested",
      description: "Your request for access to Statistical Data has been sent to the
admin.",
    });
  };

  const filterDataByTimeRange = (data: any[]) => {
    if (data.length === 0) return [];

    const now = new Date();
    const startDate = new Date();

    switch (timeRange) {
      case "week":
        startDate.setDate(now.getDate() - 7);
      break;
      case "month":
        startDate.setMonth(now.getMonth() - 1);
      break;
      case "year":
        startDate.setFullYear(now.getFullYear() - 1);
      break;
      default: // day
        startDate.setDate(now.getDate() - 1);
    }

    return data.filter((item: any) => new Date(item.lastStatusChange) >= startDate);
  };

  const calculateRoomUsage = (roomName: string): number => {
    return devices
    .filter((device) => device.room === roomName)
    .reduce((total, device) => total + device.totalEnergyConsumed, 0);
  };
```

```
  const deviceData =
    devices.length > 0
      ? filterDataByTimeRange(devices).map((device) => ({
        name: device.name,
        usage: Number.parseFloat(device.totalEnergyConsumed.toFixed(2)),
      }))
        : [];

        const roomData = rooms
        .map((room) => ({
          name: room.name,
          usage: Number.parseFloat(calculateRoomUsage(room.name).toFixed(2)),
        }))
        .filter((room) => room.usage > 0);

        const deviceTypeData =
          devices.length > 0
            ? filterDataByTimeRange(devices).reduce(
              (acc, device) => {
                acc[device.type] = (acc[device.type] || 0) +
device.totalEnergyConsumed;
                return acc;
              },
              {} as Record<string, number>
            )
              : {};

              const totalEnergyConsumed = devices.reduce((total, device) => total +
device.totalEnergyConsumed, 0);

              const containerVariants = {
                hidden: { opacity: 0 },
                visible: {
                  opacity: 1,
                  transition: {
                    staggerChildren: 0.1,
                  },
                },
              };

              const itemVariants = {
                hidden: { y: 20, opacity: 0 },
                visible: {
                  y: 0,
                  opacity: 1,
                },
              };

              if (!user) return null;

              if (user.type === "household" && !user.permissions?.statisticalData) {
```

```
                return (
                  <div className="min-h-screen bg-gray-50 flex items-center justify-
center">
                  <Card className="w-full max-w-md">
                  <CardContent className="flex flex-col items-center justify-center
p-6">
                  <Lock className="w-12 h-12 text-gray-400 mb-4" />
                  <h3 className="text-lg font-semibold mb-2">Access Required</h3>
                  <p className="text-sm text-gray-500 text-center mb-4">
                  You don't have permission to view statistical data.
                    </p>
                  <Button onClick={handleRequestAccess}>Request Access</Button>
                  </CardContent>
                  </Card>
                  </div>
                );
              }

              return (
                <div className="min-h-screen bg-gray-50 flex">
                <NavigationSidebar />
                <div className="flex-1 ml-[72px]">
                <motion.div
                initial="hidden"
                animate="visible"
                variants={containerVariants}
                className="p-6"
                >
                <motion.header variants={itemVariants} className="flex justify-
between items-center mb-8">
                <div className="flex items-center gap-4">
                <div className="w-12 h-12 bg-gradient-to-br from-[#00B2FF] to-
[#0085FF] rounded-full flex items-center justify-center shadow-lg">
                <BarChart2 className="h-6 w-6 text-white" />
                </div>
                <div>
                <h1 className="text-2xl font-bold text-gray-800">Energy
Statistics</h1>
                <p className="text-sm text-gray-500">Monitor and analyze your energy
consumption</p>
                </div>
                </div>
                <div className="flex items-center gap-4">
                <Select value={timeRange} onValueChange={setTimeRange}>
                <SelectTrigger className="w-[180px] border-[#00B2FF] text-
[#00B2FF]">
                <SelectValue placeholder="Select time range" />
                </SelectTrigger>
                <SelectContent>
                <SelectItem value="day">Past 24 Hours</SelectItem>
                <SelectItem value="week">Past Week</SelectItem>
                <SelectItem value="month">Past Month</SelectItem>
```

```
                <SelectItem value="year">Past Year</SelectItem>
                </SelectContent>
                </Select>

                {/* Dialog for Enhanced Report Download */}
                <Dialog open={isReportDialogOpen} onOpenChange=
{setIsReportDialogOpen}>
                <DialogTrigger asChild>
                <Button
                variant="outline"
                size="icon"
                className="text-[#FF9500] border-[#FF9500] hover:bg-[#FF9500]
hover:text-white"
                >
                <Download className="h-5 w-5" />
                </Button>
                </DialogTrigger>
                <DialogContent className="sm:max-w-md">
                <DialogHeader>
                <DialogTitle>Download Energy Report</DialogTitle>
                <DialogDescription>
                Choose the format for your energy consumption report
                  </DialogDescription>
                </DialogHeader>
                <div className="grid grid-cols-2 gap-4 py-4">
                <Button
                onClick={() => {
                  downloadEnhancedReport('pdf', timeRange, toast);
                  setIsReportDialogOpen(false);
                }}
                className="flex flex-col items-center justify-center h-24 p-4 border
rounded-lg hover:bg-blue-50"
                variant="outline"
                >
                <FileText className="h-8 w-8 mb-2 text-blue-600" />
                <span>PDF Report</span>
                <span className="text-xs text-gray-500">Rich visualizations</span>
                </Button>
                <Button
                onClick={() => {
                  downloadEnhancedReport('csv', timeRange, toast);
                  setIsReportDialogOpen(false);
                }}
                className="flex flex-col items-center justify-center h-24 p-4 border
rounded-lg hover:bg-green-50"
                variant="outline"
                >
                <BarChart2 className="h-8 w-8 mb-2 text-green-600" />
                <span>CSV Data</span>
                <span className="text-xs text-gray-500">For data analysis</span>
                </Button>
                </div>
```

```
                <DialogFooter className="sm:justify-between">
                <Button
                variant="ghost"
                onClick={() => {
                  generatePDF(devices, rooms);
                  setIsReportDialogOpen(false);
                }}
                className="text-gray-500"
                >
                Use Legacy Report
                </Button>
                <DialogClose asChild>
                <Button type="button" variant="secondary">
                Cancel
                </Button>
                </DialogClose>
                </DialogFooter>
                </DialogContent>
                </Dialog>

                <Button
                variant="outline"
                size="icon"
                className="text-[#00B2FF] border-[#00B2FF] hover:bg-[#00B2FF]
hover:text-white"
                >
                <User className="h-5 w-5" />
                </Button>
                </div>
                </motion.header>

                <motion.div variants={containerVariants} className="grid grid-cols-1
md:grid-cols-3 gap-6 mb-8">
                <motion.div variants={itemVariants}>
                <Card className="bg-gradient-to-br from-[#00B2FF] to-[#0085FF] text-
white">
                <CardHeader>
                <CardTitle className="text-lg font-medium">Total Energy
Consumed</CardTitle>
                </CardHeader>
                <CardContent>
                <div className="text-3xl font-bold">{totalEnergyConsumed.toFixed(2)}
kWh</div>
                <p className="text-sm opacity-80">in the selected time range</p>
                </CardContent>
                </Card>
                </motion.div>
                <motion.div variants={itemVariants}>
                <Card className="bg-gradient-to-br from-[#FF9500] to-[#FFB800] text-
white">
                <CardHeader>
                <CardTitle className="text-lg font-medium">Active
```

```
Devices</CardTitle>
                </CardHeader>
                <CardContent>
                <div className="text-3xl font-bold">{devices.filter((d) => d.status
=== "on").length}</div>
                <p className="text-sm opacity-80">out of {devices.length} total
devices</p>
                </CardContent>
                </Card>
                </motion.div>
                <motion.div variants={itemVariants}>
                <Card className="bg-gradient-to-br from-[#00B2FF] to-[#0085FF] text-
white">
                <CardHeader>
                <CardTitle className="text-lg font-medium">Most Active
Room</CardTitle>
                </CardHeader>
                <CardContent>
                <div className="text-3xl font-bold">
                {roomData.length > 0 ? roomData.reduce((a, b) => (a.usage > b.usage
? a : b)).name : "No data"}
                </div>
                <p className="text-sm opacity-80">based on energy consumption</p>
                </CardContent>
                </Card>
                </motion.div>
                </motion.div>

                <motion.div variants={itemVariants}>
                <Tabs defaultValue="devices" className="space-y-6">
                <TabsList className="bg-white shadow-md rounded-lg p-1">
                <TabsTrigger value="devices" className="data-[state=active]:bg-
[#00B2FF] data-[state=active]:text-white">
                <BarChart2 className="h-4 w-4 mr-2" />
                Devices
                </TabsTrigger>
                <TabsTrigger value="rooms" className="data-[state=active]:bg-
[#00B2FF] data-[state=active]:text-white">
                <PieChartIcon className="h-4 w-4 mr-2" />
                Rooms
                </TabsTrigger>
                <TabsTrigger value="overview" className="data-[state=active]:bg-
[#00B2FF] data-[state=active]:text-white">
                <TrendingUp className="h-4 w-4 mr-2" />
                Overview
                </TabsTrigger>
                </TabsList>
                <TabsContent value="devices" className="space-y-6">
                <Card>
                <CardHeader>
                <CardTitle className="text-xl font-semibold text-gray-800">Energy
Usage by Device</CardTitle>
```

```
                </CardHeader>
                <CardContent>
                <div className="chart-container">
                {deviceData.length > 0 ? (
                  <BarChart data={deviceData} />
                ) : (
                <p className="text-center text-gray-500">No data available for the
selected time range</p>
                )}
              </div>
              </CardContent>
              </Card>
              <Card>
              <CardHeader>
              <CardTitle className="text-xl font-semibold text-gray-800">Device
Usage Over Time</CardTitle>
              </CardHeader>
              <CardContent>
              <div className="chart-container">
              {deviceData.length > 0 ? (
                <LineChart data={deviceData} />
              ) : (
              <p className="text-center text-gray-500">No data available for the
selected time range</p>
              )}
            </div>
            </CardContent>
            </Card>
            </TabsContent>
            <TabsContent value="rooms" className="space-y-6">
            <Card>
            <CardHeader>
            <CardTitle className="text-xl font-semibold text-gray-800">Energy Usage
by Room</CardTitle>
            </CardHeader>
            <CardContent>
            <div className="chart-container">
            {roomData.length > 0 ? (
              <BarChart data={roomData} />
            ) : (
            <p className="text-center text-gray-500">No data available for the
selected time range</p>
            )}
          </div>
          </CardContent>
          </Card>
          <Card>
          <CardHeader>
          <CardTitle className="text-xl font-semibold text-gray-800">Room Usage Over
Time</CardTitle>
          </CardHeader>
          <CardContent>
```

```
          <div className="chart-container">
          {roomData.length > 0 ? (
            <LineChart data={roomData} />
          ) : (
          <p className="text-center text-gray-500">No data available for the
selected time range</p>
          )}
        </div>
        </CardContent>
        </Card>
        </TabsContent>
        <TabsContent value="overview" className="space-y-6">
        <Card>
        <CardHeader>
        <CardTitle className="text-xl font-semibold text-gray-800">Energy Usage by
Device Type</CardTitle>
        </CardHeader>
        <CardContent>
        <div className="chart-container">
        {Object.keys(deviceTypeData).length > 0 ? (
          <PieChart
          data={Object.entries(deviceTypeData).map(([name, usage]) => ({
            name,
            usage: Number.parseFloat(usage.toFixed(2)),
          }))}
          />
        ) : (
        <p className="text-center text-gray-500">No data available for the selected
time range</p>
        )}
      </div>
      </CardContent>
      </Card>
      <Card>
      <CardHeader>
      <CardTitle className="text-xl font-semibold text-gray-800">Total Energy
Consumption Trend</CardTitle>
      </CardHeader>
      <CardContent>
      <div className="chart-container">
      {deviceData.length > 0 ? (
        <LineChart data={deviceData} />
      ) : (
      <p className="text-center text-gray-500">No data available for the selected
time range</p>
      )}
    </div>
    </CardContent>
    </Card>
    </TabsContent>
    </Tabs>
    </motion.div>
```

```
    </motion.div>
    </div>
    </div>
              );
}
```

app/suggestions/page.tsx

```
// suggestions/page.tsx
"use client"

import { useState, useEffect } from "react"
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/card"
import { Button } from "@/components/ui/button"
import { Badge } from "@/components/ui/badge"
import { Download, User, Lightbulb, ThermometerSun, Timer, Fan, ArrowRight, Zap }
from "lucide-react"
import { motion } from "framer-motion"
import { NavigationSidebar } from "@/app/components/navigation-sidebar";

const adminSuggestions = [
  {
    title: "Optimize Air Conditioning",
    description: "Your bedroom AC has been running continuously for 8 hours. Setting
up a schedule could save energy.",
    icon: Fan,
    iconColor: "text-blue-500",
    saving: "Potential saving: 120 kWh/month",
    impact: "Environmental Impact: Reduce CO2 emissions by 60kg",
    action: "Setup Schedule",
    details: [
      "Current Usage: 8 hours continuous operation",
      "Recommended: 15-minute intervals with 2°C adjustment",
      "Peak hours detected: 2 PM - 5 PM",
    ],
    category: "Energy Efficiency",
  },
  {
    title: "Smart Lighting Pattern Detected",
    description: "We noticed regular lighting patterns in your living room. Would
you like to automate this?",
    icon: Lightbulb,
    iconColor: "text-yellow-500",
    saving: "Potential saving: 45 kWh/month",
    impact: "Convenience: Automate daily routines",
    action: "Create Automation",
    details: [
      "Pattern detected: Lights on 6 PM - 11 PM",
      "Affects: Living Room, Kitchen",
      "Suggested: Gradual dimming by time",
```

```
    ],
    category: "Automation",
  },
  {
    title: "Temperature Optimization",
    description: "Your home's temperature varies significantly throughout the day.
Let's optimize it.",
    icon: ThermometerSun,
    iconColor: "text-red-500",
    saving: "Potential saving: AED 150/month",
    impact: "Comfort: Maintain optimal temperature",
    action: "View Analysis",
    details: ["Current range: 19°C - 26°C", "Recommended: 23°C - 24°C", "Affected
rooms: Bedroom, Living Room"],
    category: "Energy Efficiency",
  },
  {
    title: "Off-Peak Usage Opportunity",
    description: "Running your washing machine during off-peak hours could reduce
your electricity bill.",
    icon: Timer,
    iconColor: "text-green-500",
    saving: "Potential saving: AED 80/month",
    impact: "Grid Impact: Reduce peak load stress",
    action: "See Off-Peak Hours",
    details: ["Peak hours: 2 PM - 6 PM", "Off-peak rates: 11 PM - 5 AM", "Applicable
to: Washing Machine, Dishwasher"],
    category: "Cost Saving",
  },
]

const householdSuggestions = [
  {
    title: "Room Temperature Alert",
    description: "Your room temperature is set to 19°C. Consider increasing it to
save energy.",
    icon: ThermometerSun,
    iconColor: "text-red-500",
    saving: "Potential saving: 30 kWh/month",
    impact: "Personal Comfort: Maintain optimal temperature",
    action: "Adjust Temperature",
    details: ["Current: 19°C", "Recommended: 23°C", "Energy usage: Above average"],
    category: "Energy Efficiency",
  },
  {
    title: "Lighting Reminder",
    description: "Lights in your room are often left on when you're away.",
    icon: Lightbulb,
    iconColor: "text-yellow-500",
    saving: "Potential saving: 15 kWh/month",
    impact: "Easy Fix: Use motion sensors",
    action: "Enable Auto-off",
```

```
    details: ["Pattern: Lights left on 2+ hours", "Suggestion: Enable motion
detection", "Applies to: Your bedroom"],
    category: "Energy Efficiency",
  },
]

export default function SuggestionsPage() {
  const [suggestions, setSuggestions] = useState(adminSuggestions)
  const [userType, setUserType] = useState<"admin" | "household">("admin")

  useEffect(() => {
    const currentUser = JSON.parse(localStorage.getItem("currentUser") || '{"type":
"admin"}')
    setUserType(currentUser.type)
    setSuggestions(currentUser.type === "admin" ? adminSuggestions :
householdSuggestions)
  }, [])

  const containerVariants = {
    hidden: { opacity: 0 },
    visible: {
      opacity: 1,
      transition: {
        staggerChildren: 0.1,
      },
    },
  }

  const itemVariants = {
    hidden: { y: 20, opacity: 0 },
    visible: {
      y: 0,
      opacity: 1,
    },
  }

  return (
    <div className="min-h-screen bg-gray-50">
      <NavigationSidebar /> {/* Add the navbar here */}
      <div className="ml-[72px]">
      <motion.div initial="hidden" animate="visible" variants={containerVariants}
className="p-6">
        <motion.header variants={itemVariants} className="flex justify-between
items-center mb-6">
          <div className="flex items-center gap-4">
            <div className="w-12 h-12 bg-gradient-to-br from-[#00B2FF] to-[#0085FF]
rounded-full flex items-center justify-center shadow-lg">
              <Zap className="h-6 w-6 text-white" />
            </div>
            <div>
              <h1 className="text-2xl font-bold text-gray-800">Smart
Suggestions</h1>
```

```
              <p className="text-sm text-gray-500">Optimize your home's energy
usage</p>
            </div>
          </div>
          <div className="flex gap-2">
            <Button
              variant="outline"
              size="icon"
              className="text-[#00B2FF] border-[#00B2FF] hover:bg-[#00B2FF]
hover:text-white"
            >
              <Download className="h-5 w-5" />
            </Button>
            <Button
              variant="outline"
              size="icon"
              className="text-[#00B2FF] border-[#00B2FF] hover:bg-[#00B2FF]
hover:text-white"
            >
              <User className="h-5 w-5" />
            </Button>
          </div>
        </motion.header>

        <motion.div variants={containerVariants} className="grid grid-cols-1
md:grid-cols-2 gap-6">
          {suggestions.map((suggestion, index) => (
            <motion.div key={suggestion.title} variants={itemVariants}>
              <Card className="overflow-hidden hover:shadow-lg transition-shadow
duration-300">
                <CardHeader className="bg-gradient-to-r from-[#00B2FF] to-[#0085FF]
pb-8 relative">
                  <CardTitle className="text-white text-xl mb-2">{suggestion.title}
</CardTitle>
                  <p className="text-white/80 text-sm">{suggestion.description}</p>
                  <div className="absolute bottom-0 right-0 transform translate-y-
1/2 mr-6">
                    <div className="w-12 h-12 rounded-full bg-white flex items-
center justify-center shadow-lg">
                      <suggestion.icon className={`w-6 h-6 ${suggestion.iconColor}`}
/>
                    </div>
                  </div>
                </CardHeader>
                <CardContent className="pt-8">
                  <div className="space-y-4">
                    <div>
                      <Badge variant="secondary" className="mb-2">
                        {suggestion.category}
                      </Badge>
                      <p className="text-sm font-medium text-green-600">
{suggestion.saving}</p>
```

```
                    <p className="text-sm text-gray-600">{suggestion.impact}</p>
                  </div>
                  <div className="space-y-2">
                    {suggestion.details.map((detail, idx) => (
                      <div key={idx} className="flex items-center text-sm text-
gray-600">
                        <div className="w-1.5 h-1.5 rounded-full bg-[#00B2FF] mr-
2" />
                        {detail}
                      </div>
                    ))}
                  </div>
                  <Button className="w-full bg-[#00B2FF] hover:bg-[#00B2FF]/90">
                    {suggestion.action}
                    <ArrowRight className="ml-2 h-4 w-4" />
                  </Button>
                </div>
              </CardContent>
            </Card>
          </motion.div>
        ))}
      </motion.div>
    </motion.div>
      </div>
  </div>
  )
}
```