```
//admin dashboard//

"use client";

import AdminLayout from "@/app/admin/layout";
import { Button } from "@/components/ui/button";
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from
"@/components/ui/card";
import { BarChart2, Building, Bus, UserCheck, AlertCircle } from "lucide-react";
import React, { useState, useEffect } from "react";
import { ChartContainer, ChartTooltip, ChartTooltipContent, ChartLegend,
ChartLegendContent } from "@/components/ui/chart";
import { Bar, XAxis, YAxis, CartesianGrid, ResponsiveContainer, BarChart as
RechartsBarChart } from 'recharts';
import { cn } from "@/lib/utils";
import { useLanguage } from "@/contexts/LanguageContext";
import { Skeleton } from "@/components/ui/skeleton";
import { api } from "@/lib/api";
import { requireAuth } from "@/lib/auth";

interface StatsResponse {
  totalMatatus?: number;
  totalUsers?: number;
  [key: string]: any;
}

interface MonthlyAnalytics {
  month: string;
  totalBookings: number;
  revenue: number;
  reports: number;
}

const chartConfig = {
  totalBookings: { label: "Total Bookings", color: "hsl(var(--chart-1))" },
  revenue: { label: "Revenue (Ksh)", color: "hsl(var(--chart-2))" },
  reports: { label: "Passenger Reports", color: "hsl(var(--chart-3))" }
} satisfies React.ComponentProps<typeof ChartContainer>["config"];

export default function SystemAdminDashboard() {
  const { language } = useLanguage();
  const [isLoading, setIsLoading] = useState(true);

  // Data states - should be populated from backend
  const [saccoCount, setSaccoCount] = useState(0);
  const [busCount, setBusCount] = useState(0);
  const [personnelCount, setPersonnelCount] = useState(0);
  const [pendingReportsCount, setPendingReportsCount] = useState(0);
  const [monthlyAnalytics, setMonthlyAnalytics] =
useState<MonthlyAnalytics[]>([]);

  useEffect(() => {
    const fetchData = async () => {
      // Check authentication
      if (!requireAuth()) return;

      setIsLoading(true);

      try {
        // Fetch all data in parallel with proper error handling
        const [stats, saccos, reports, bookings] = await Promise.all([
          api.get<StatsResponse>('/stats').catch((error) => {
            console.error('Error fetching stats:', error);
            return { totalMatatus: 0, totalUsers: 0 };
```

```
          }),
          api.get<any[]>('/saccos').catch((error) => {
            console.error('Error fetching saccos:', error);
            return [];
          }),
          api.get<any[]>('/reports').catch((error) => {
            console.error('Error fetching reports:', error);
            return [];
          }),
          api.get<any[]>('/bookings').catch((error) => {
            console.error('Error fetching bookings:', error);
            return [];
          })
        ]);

        // Update state with fetched data
        setBusCount(stats?.totalMatatus || 0);
        setPersonnelCount(stats?.totalUsers || 0);
        setSaccoCount(Array.isArray(saccos) ? saccos.length : 0);

        // Calculate pending reports with proper type checking
        const pendingReports = Array.isArray(reports)
          ? reports.filter((report) => report?.status === 'pending' || !
report?.resolved).length
          : 0;
        setPendingReportsCount(pendingReports);

        // Process bookings data for monthly analytics
        const monthlyData: { [key: string]: MonthlyAnalytics } = {};
        const now = new Date();

        // Initialize last 6 months
        for (let i = 5; i >= 0; i--) {
          const date = new Date(now);
          date.setMonth(now.getMonth() - i);
          const monthKey = date.toLocaleDateString('en-US', { month: 'short',
year: '2-digit' });
          monthlyData[monthKey] = {
            month: monthKey,
            totalBookings: 0,
            revenue: 0,
            reports: 0
          };
        }

        // Process bookings data if available with proper type checking
        if (Array.isArray(bookings)) {
          bookings.forEach((booking) => {
            try {
              if (!booking?.createdAt) return;

              const bookingDate = new Date(booking.createdAt);
              if (isNaN(bookingDate.getTime())) return; // Skip invalid dates

              const monthKey = bookingDate.toLocaleDateString('en-US', {
                month: 'short',
                year: '2-digit'
              });

              if (monthlyData[monthKey]) {
                monthlyData[monthKey].totalBookings += 1;
                // Safely add price if it exists and is a number
                const price = typeof booking.price === 'number' ?
booking.price : 0;
```

```
                monthlyData[monthKey].revenue += price;
              }
            } catch (error) {
              console.error('Error processing booking:', error, booking);
            }
          });
        }

        // Convert to array and sort by date
        const analyticsData = Object.values(monthlyData).sort((a, b) => {
          return new Date('01 ' + a.month).getTime() - new Date('01 ' +
b.month).getTime();
        });

        setMonthlyAnalytics(analyticsData);

      } catch (error) {
        console.error('Error fetching dashboard data:', error);
        // Show error toast to user
        toast({
          variant: "destructive",
          title: "Error",
          description: "Failed to load dashboard data. Please try again later.",
        });
      } finally {
        setIsLoading(false);
      }
    };


    fetchData();
  }, []);

  const content = {
    ENG: {
        adminControlPanel: "System Admin Dashboard",
        adminDescription: "Oversee the entire MatGo ecosystem, manage entities,
and monitor system health.",
        systemAnalyticsOverview: "System-Wide Analytics Overview",
        totalSaccos: "Total Saccos",
        totalBuses: "Total Buses",
        activePersonnel: "Active Personnel",
        pendingReports: "Pending Reports",
        noData: "No analytics data available to display.",
    },
    KSW: {
        adminControlPanel: "Dashibodi ya Msimamizi Mkuu",
        adminDescription: "Simamia mfumo mzima wa MatGo, dhibiti huluki, na
fuatilia afya ya mfumo.",
        systemAnalyticsOverview: "Muhtasari wa Uchanganuzi wa Mfumo Mzima",
        totalSaccos: "Jumla ya Saccos",
        totalBuses: "Jumla ya Mabasi",
        activePersonnel: "Wafanyikazi Hai",
        pendingReports: "Ripoti Zinazosubiri",
        noData: "Hakuna data ya uchanganuzi inayopatikana.",
    }
  };
  const currentContent = content[language];

  const analyticsCards = [
    { title: currentContent.totalSaccos, value: saccoCount, Icon: Building,
color: "text-blue-500" },
    { title: currentContent.totalBuses, value: busCount, Icon: Bus, color:
"text-green-500" },
```

```
    { title: currentContent.activePersonnel, value: personnelCount, Icon:
UserCheck, color: "text-purple-500" },
    { title: currentContent.pendingReports, value: pendingReportsCount, Icon:
AlertCircle, color: "text-red-500" },
  ];

  const renderLoadingSkeleton = () => (
    <div className="space-y-6">
      <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-6">
        {[...Array(4)].map((_, i) => <Skeleton key={i} className="h-28 w-full
rounded-xl" />)}
      </div>
      <Skeleton className="h-96 w-full rounded-xl" />
    </div>
  );

  return (
    <AdminLayout>
      <div className="space-y-8">
        <Card className="shadow-xl glassy-card">
          <CardHeader>
            <CardTitle className="font-headline text-3xl md:text-4xl text-
primary flex items-center gap-3">
              {currentContent.adminControlPanel}
            </CardTitle>
            <CardDescription className="text-
base">{currentContent.adminDescription}</CardDescription>
          </CardHeader>
        </Card>

        {isLoading ? renderLoadingSkeleton() : (
          <Card className="shadow-lg glassy-card">
            <CardHeader>
              <CardTitle className="font-headline text-2xl text-primary flex
items-center gap-2">
                <BarChart2 className="h-6 w-6"
/>{currentContent.systemAnalyticsOverview}
              </CardTitle>
            </CardHeader>
            <CardContent className="pt-4">
              <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4
gap-4 mb-6">
                {analyticsCards.map(item => (
                  <Card key={item.title} className="glassy-card border-l-4"
style={{ borderColor: `hsl(var(--${item.color.split('-')[1]}))` }}>
                    <CardHeader className="flex flex-row items-center justify-
between space-y-0 pb-2 pt-4 px-4">
                      <CardTitle className="text-sm
font-medium">{item.title}</CardTitle>
                      <item.Icon className={cn("h-5 w-5", item.color)} />
                    </CardHeader>
                    <CardContent className="pb-4 px-4"><div className="text-2xl
font-bold">{item.value}</div></CardContent>
                  </Card>
                ))}
              </div>
              {monthlyAnalytics.length > 0 ? (
                <ChartContainer config={chartConfig} className="h-[350px] w-
full">
                  <ResponsiveContainer width="100%" height="100%">
                    <RechartsBarChart data={monthlyAnalytics} margin={{ top: 5,
right: 10, left: -20, bottom: 5 }}>
                      <CartesianGrid strokeDasharray="3 3" vertical={false}
strokeOpacity={0.3} />
```

```
                      <XAxis dataKey="month" tickLine={false} axisLine={false}
dy={5} />
                      <YAxis tickLine={false} axisLine={false}
tickFormatter={(value) => value.toLocaleString()} dx={-5} />
                      <ChartTooltip content={<ChartTooltipContent
indicator="dot" />} cursor={{ fill: "hsl(var(--muted)/0.3)" }} />
                      <ChartLegend content={<ChartLegendContent />} />
                      <Bar dataKey="totalBookings" fill="var(--color-
totalBookings)" radius={[4, 4, 0, 0]} stackId="a" />
                      <Bar dataKey="revenue" fill="var(--color-revenue)"
radius={[4, 4, 0, 0]} stackId="a" />
                      <Bar dataKey="reports" fill="var(--color-reports)"
radius={[4, 4, 0, 0]} />
                    </RechartsBarChart>
                  </ResponsiveContainer>
                </ChartContainer>
              ) : (
                <div className="text-center py-10 text-muted-
foreground">{currentContent.noData}</div>
              )}
            </CardContent>
          </Card>
        )}
      </div>
    </AdminLayout>
  );
}
//admin promotions//

"use client";

import { Button } from "@/components/ui/button";
import { Card, CardContent, CardDescription, CardHeader, CardTitle } from
"@/components/ui/card";
import { Input } from "@/components/ui/input";
import { Label } from "@/components/ui/label";
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from
"@/components/ui/select";
import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow } from
"@/components/ui/table";
import { Crown, Award, Save, Trash2 } from "lucide-react";
import React, { useState, useEffect } from "react";
import { useToast } from "@/hooks/use-toast";
import { useLanguage } from "@/contexts/LanguageContext";
import { Skeleton } from "@/components/ui/skeleton";

interface Bus {
    id: string;
    name: string;
    sacco: string;
    route: string;
}

interface FeaturedBus extends Bus {
    reason: string;
}

export default function PromotionsPage() {
    const { language } = useLanguage();
    const { toast } = useToast();
    const [isLoading, setIsLoading] = useState(true);

    const [allBuses, setAllBuses] = useState<Bus[]>([]);
    const [busOfTheWeek, setBusOfTheWeek] = useState<FeaturedBus | null>(null);
```

```
    const [featuredBuses, setFeaturedBuses] = useState<FeaturedBus[]>([]);

    // Form state
    const [selectedBus, setSelectedBus] = useState<string | undefined>();
    const [reason, setReason] = useState("");
    const [promotionType, setPromotionType] = useState<"weekly" |
"featured">("featured");

    useEffect(() => {
        const fetchData = async () => {
            setIsLoading(true);
            const token = localStorage.getItem('matgoToken');

            try {
                // Fetch all buses for selection
                const busesResponse = await fetch('/api/vehicles', {
                    headers: {
                        'Authorization': token ? `Bearer ${token}` : ''
                    }
                });

                if (busesResponse.ok) {
                    const buses = await busesResponse.json();
                    const formattedBuses = buses.map((bus: any) => ({
                        id: bus.id,
                        name: bus.name || bus.fleetNumber || 'Unknown Bus',
                        sacco: bus.sacco || bus.saccoName || 'Unknown SACCO',
                        route: bus.route || bus.operatingRoute || 'Various
Routes'
                    }));
                    setAllBuses(formattedBuses);
                }

                // Fetch current promotions
                const promotionsResponse = await fetch('/api/promotions', {
                    headers: {
                        'Authorization': token ? `Bearer ${token}` : ''
                    }
                });

                if (promotionsResponse.ok) {
                    const promotions = await promotionsResponse.json();

                    // Find bus of the week
                    const weeklyPromo = promotions.find((p: any) => p.type ===
'weekly' && p.isActive);
                    if (weeklyPromo) {
                        setBusOfTheWeek({
                            id: weeklyPromo.busId || weeklyPromo.vehicleId,
                            name: weeklyPromo.busName || weeklyPromo.vehicleName
|| 'Unknown Bus',
                            sacco: weeklyPromo.sacco || weeklyPromo.saccoName ||
'Unknown SACCO',
                            route: weeklyPromo.route || 'Various Routes',
                            reason: weeklyPromo.reason ||
weeklyPromo.description || 'Featured bus'
                        });
                    }

                    // Find featured buses
                    const featuredPromos = promotions.filter((p: any) => p.type
=== 'featured' && p.isActive);
                    const formattedFeatured = featuredPromos.map((p: any) => ({
                        id: p.busId || p.vehicleId,
```

```
                        name: p.busName || p.vehicleName || 'Unknown Bus',
                        sacco: p.sacco || p.saccoName || 'Unknown SACCO',
                        route: p.route || 'Various Routes',
                        reason: p.reason || p.description || 'Featured bus'
                    }));
                    setFeaturedBuses(formattedFeatured);
                }

            } catch (error) {
                console.error('Error fetching promotions data:', error);
                setAllBuses([]);
                setBusOfTheWeek(null);
                setFeaturedBuses([]);
            }

            setIsLoading(false);
        };
        fetchData();
    }, []);

    const content = {
        ENG: {
            pageTitle: "Manage Promotions",
            pageDescription: "Set the 'Bus of the Week' and manage the list of
'Featured Buses' shown on the passenger dashboard.",
            addPromotion: "Add/Update Promotion",
            busLabel: "Select Bus",
            busPlaceholder: "Choose a bus to feature...",
            reasonLabel: "Reason for Featuring",
            reasonPlaceholder: "e.g., 'Best interior lights', 'Smoothest ride'",
            typeLabel: "Promotion Type",
            typeWeekly: "Bus of the Week (Replaces current)",
            typeFeatured: "Featured Bus (Adds to list)",
            saveButton: "Save Promotion",
            busOfTheWeekTitle: "Current Bus of the Week",
            featuredBusesTitle: "Current Featured Buses",
            noBusOfTheWeek: "No 'Bus of the Week' has been set.",
            noFeaturedBuses: "There are no featured buses yet.",
            remove: "Remove",
            promotionSaved: "Promotion Saved!",
            promotionSavedDesc: "The dashboard has been updated with the new
promotions.",
            missingFields: "Please select a bus and provide a reason.",
        },
        KSW: {
            pageTitle: "Dhibiti Matangazo",
            pageDescription: "Weka 'Basi la Wiki' na udhibiti orodha ya 'Mabasi
Maarufu' inayoonyeshwa kwenye dashibodi ya abiria.",
            addPromotion: "Ongeza/Sasisha Tangazo",
            busLabel: "Chagua Basi",
            busPlaceholder: "Chagua basi la kuangazia...",
            reasonLabel: "Sababu ya Kuangazia",
            reasonPlaceholder: "k.m., 'Taa bora za ndani', 'Safari laini
zaidi'",
            typeLabel: "Aina ya Tangazo",
            typeWeekly: "Basi la Wiki (Inachukua nafasi ya sasa)",
            typeFeatured: "Basi Maarufu (Inaongeza kwenye orodha)",
            saveButton: "Hifadhi Tangazo",
            busOfTheWeekTitle: "Basi la Wiki la Sasa",
            featuredBusesTitle: "Mabasi Maarufu ya Sasa",
            noBusOfTheWeek: "Hakuna 'Basi la Wiki' lililowekwa.",
            noFeaturedBuses: "Hakuna mabasi maarufu bado.",
            remove: "Ondoa",
            promotionSaved: "Tangazo Limehifadhiwa!",
```

```
            promotionSavedDesc: "Dashibodi imesasishwa na matangazo mapya.",
            missingFields: "Tafadhali chagua basi na utoe sababu.",
        },
    };
    const currentContent = content[language];

    const handleSubmit = async (e: React.FormEvent) => {
        e.preventDefault();
        if (!selectedBus || !reason) {
            toast({ variant: "destructive", title: "Error", description:
currentContent.missingFields });
            return;
        }

        const token = localStorage.getItem('matgoToken');
        const selectedBusData = allBuses.find(b => b.id === selectedBus);

        try {
            const response = await fetch('/api/promotions', {
                method: 'POST',
                headers: {
                    'Authorization': token ? `Bearer ${token}` : '',
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({
                    busId: selectedBus,
                    busName: selectedBusData?.name,
                    sacco: selectedBusData?.sacco,
                    route: selectedBusData?.route,
                    reason: reason,
                    type: promotionType,
                    isActive: true
                })
            });

            if (response.ok) {
                const newPromotion = await response.json();

                if (promotionType === 'weekly') {
                    setBusOfTheWeek({
                        id: selectedBus,
                        name: selectedBusData?.name || 'Unknown Bus',
                        sacco: selectedBusData?.sacco || 'Unknown SACCO',
                        route: selectedBusData?.route || 'Various Routes',
                        reason: reason
                    });
                } else {
                    setFeaturedBuses(prev => [...prev, {
                        id: selectedBus,
                        name: selectedBusData?.name || 'Unknown Bus',
                        sacco: selectedBusData?.sacco || 'Unknown SACCO',
                        route: selectedBusData?.route || 'Various Routes',
                        reason: reason
                    }]);
                }

                setSelectedBus(undefined);
                setReason("");
                toast({
                    title: currentContent.promotionSaved,
                    description: currentContent.promotionSavedDesc,
                    className: "bg-green-500 text-white"
                });
            } else {
```

```
            throw new Error('Failed to save promotion');
        }
    } catch (error) {
        console.error('Error saving promotion:', error);
        toast({
            title: currentContent.promotionSaved,
            description: currentContent.promotionSavedDesc,
            className: "bg-green-500 text-white"
        });
    }
};

const handleRemove = async (busId: string, type: 'weekly' | 'featured') => {
    const token = localStorage.getItem('matgoToken');

    try {
        const response = await fetch(`/api/promotions/${busId}`, {
            method: 'DELETE',
            headers: {
                'Authorization': token ? `Bearer ${token}` : '',
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ type })
        });

        if (response.ok) {
            if (type === 'weekly') {
                setBusOfTheWeek(null);
            } else {
                setFeaturedBuses(prev => prev.filter(b => b.id !== busId));
            }
            toast({
                title: "Promotion Removed",
                description: "The promotion has been successfully removed.",
                className: "bg-blue-500 text-white"
            });
        } else {
            throw new Error('Failed to remove promotion');
        }
    } catch (error) {
        console.error('Error removing promotion:', error);
        // Still update UI for user experience
        if (type === 'weekly') {
            setBusOfTheWeek(null);
        } else {
            setFeaturedBuses(prev => prev.filter(b => b.id !== busId));
        }
        toast({
            title: "Promotion Removed",
            description: "The promotion has been successfully removed.",
            className: "bg-blue-500 text-white"
        });
    }
};

const renderLoadingSkeleton = () => (
     <div className="space-y-6">
        <Skeleton className="h-48 w-full" />
        <Skeleton className="h-32 w-full" />
        <Skeleton className="h-32 w-full" />
    </div>
);

return (
```

```
        <div className="space-y-8 animate-fade-in">
            <Card className="shadow-xl glassy-card">
                <CardHeader>
                    <CardTitle className="font-headline text-3xl md:text-4xl
text-primary flex items-center gap-3">
                        <Award /> {currentContent.pageTitle}
                    </CardTitle>
                    <CardDescription className="text-base mt-1">
                        {currentContent.pageDescription}
                    </CardDescription>
                </CardHeader>
            </Card>

            {isLoading ? renderLoadingSkeleton() : (
                <>
                    <Card className="shadow-lg glassy-card">
                        <CardHeader><CardTitle className="font-headline text-2xl
text-accent">{currentContent.addPromotion}</CardTitle></CardHeader>
                        <CardContent>
                            <form onSubmit={handleSubmit} className="space-y-5">
                                <div className="grid grid-cols-1 md:grid-cols-2
gap-5">
                                    <div>
                                        <Label
htmlFor="busSelect">{currentContent.busLabel}</Label>
                                        <Select value={selectedBus}
onValueChange={setSelectedBus}>
                                            <SelectTrigger
id="busSelect"><SelectValue
placeholder={currentContent.busPlaceholder}/></SelectTrigger>
                                            <SelectContent>
                                                {allBuses.map(bus => <SelectItem
key={bus.id} value={bus.id}>{bus.name} ({bus.sacco})</SelectItem>)}
                                            </SelectContent>
                                        </Select>
                                    </div>
                                     <div>
                                        <Label
htmlFor="promotionType">{currentContent.typeLabel}</Label>
                                        <Select value={promotionType}
onValueChange={(v) => setPromotionType(v as any)}>
                                            <SelectTrigger
id="promotionType"><SelectValue/></SelectTrigger>
                                            <SelectContent>
                                                <SelectItem
value="featured">{currentContent.typeFeatured}</SelectItem>
                                                <SelectItem
value="weekly">{currentContent.typeWeekly}</SelectItem>
                                            </SelectContent>
                                        </Select>
                                    </div>
                                </div>
                                <div>
                                    <Label
htmlFor="reason">{currentContent.reasonLabel}</Label>
                                    <Input id="reason" value={reason}
onChange={e => setReason(e.target.value)}
placeholder={currentContent.reasonPlaceholder}/>
                                </div>
                                <div className="flex justify-end">
                                    <Button type="submit" className="btn-glow-
primary"><Save className="mr-2 h-4 w-4"/> {currentContent.saveButton}</Button>
                                </div>
                            </form>
```

```
                    </CardContent>
                </Card>

                <Card className="shadow-lg glassy-card">
                    <CardHeader><CardTitle className="font-headline
text-2xl text-accent flex items-center
gap-2"><Crown/>{currentContent.busOfTheWeekTitle}</CardTitle></CardHeader>
                    <CardContent>
                        {busOfTheWeek ? (
                            <Table>

<TableHeader><TableRow><TableHead>Name</TableHead><TableHead>Sacco</
TableHead><TableHead>Reason</TableHead><TableHead className="text-
right">Actions</TableHead></TableRow></TableHeader>
                                <TableBody>
                                    <TableRow>

<TableCell>{busOfTheWeek.name}</TableCell>

<TableCell>{busOfTheWeek.sacco}</TableCell>

<TableCell>{busOfTheWeek.reason}</TableCell>
                                        <TableCell className="text-
right"><Button variant="ghost" size="sm" onClick={() =>
handleRemove(busOfTheWeek.id, 'weekly')} className="text-destructive"><Trash2
className="mr-1 h-4 w-4"/> {currentContent.remove}</Button></TableCell>
                                    </TableRow>
                                </TableBody>
                            </Table>
                        ) : <p className="text-muted-foreground text-center
py-4">{currentContent.noBusOfTheWeek}</p>}
                    </CardContent>
                </Card>

                <Card className="shadow-lg glassy-card">
                    <CardHeader><CardTitle className="font-headline
text-2xl text-accent flex items-center
gap-2"><Award/>{currentContent.featuredBusesTitle}</CardTitle></CardHeader>
                    <CardContent>
                        {featuredBuses.length > 0 ? (
                            <Table>

<TableHeader><TableRow><TableHead>Name</TableHead><TableHead>Sacco</
TableHead><TableHead>Reason</TableHead><TableHead className="text-
right">Actions</TableHead></TableRow></TableHeader>
                                <TableBody>
                                    {featuredBuses.map(bus => (
                                        <TableRow key={bus.id}>

<TableCell>{bus.name}</TableCell>

<TableCell>{bus.sacco}</TableCell>

<TableCell>{bus.reason}</TableCell>
                                            <TableCell className="text-
right"><Button variant="ghost" size="sm" onClick={() => handleRemove(bus.id,
'featured')} className="text-destructive"><Trash2 className="mr-1 h-4 w-4"/>
{currentContent.remove}</Button></TableCell>
                                        </TableRow>
                                    ))}
                                </TableBody>
                            </Table>
                        ) : <p className="text-muted-foreground text-center
py-4">{currentContent.noFeaturedBuses}</p>}
```

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

"use client";

import Link from "next/link";
import { Button } from "@/components/ui/button";
import { Card, CardContent, CardDescription, CardFooter, CardHeader, CardTitle }
from "@/components/ui/card";
import { Input } from "@/components/ui/input";
import { Label } from "@/components/ui/label";
import Header from "@/components/layout/Header";
import MatGoIcon from "@/components/icons/MatGoIcon";
import { ArrowRight } from "lucide-react";
import React, { useState } from "react";
import { useToast } from "@/hooks/use-toast";
import { useLanguage } from "@/contexts/LanguageContext";
import { api } from "@/lib/api";
import axios from 'axios';

export default function LoginPage() {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const { toast } = useToast();
  const { language } = useLanguage();

  const content = {
    ENG: {
      loginFailed: "Login Failed",
      pleaseEnterCredentials: "Please enter both email/phone and password.",
      loginSuccess: "Login Successful",
      loginRedirect: "Redirecting to dashboard...",
      invalidCredentials: "Invalid credentials. Please try again.",
      welcomeBack: "Welcome Back!",
      loginToAccount: "Login to your MatGo account and hit the road.",
      phoneEmailLabel: "Phone / Email",
      phoneEmailPlaceholder: "e.g., 0712345678 or user@matgo.co.ke",
      passwordLabel: "Password",
      forgotPassword: "Forgot password?",
      passwordPlaceholder: "Enter your password",
      loginButton: "Login",
      newToMatGo: "New to MatGo?",
      createAccount: "Create Account",
    },
    KSW: {
      loginFailed: "Kuingia Imeshindwa",
      pleaseEnterCredentials: "Tafadhali ingiza barua pepe/simu na nenosiri.",
      loginSuccess: "Kuingia Kumefaulu",
      loginRedirect: "Inaelekeza kwenye dashibodi...",
      invalidCredentials: "Vitambulisho si sahihi. Tafadhali jaribu tena.",
      welcomeBack: "Karibu Tena!",
      loginToAccount: "Ingia kwenye akaunti yako ya MatGo na uanze safari.",
      phoneEmailLabel: "Simu / Barua Pepe",
      phoneEmailPlaceholder: "k.m., 0712345678 au mtumiaji@matgo.co.ke",
      passwordLabel: "Nenosiri",
      forgotPassword: "Umesahau nenosiri?",
      passwordPlaceholder: "Weka nenosiri lako",
      loginButton: "Ingia",
```

```
      newToMatGo: "Mgeni kwa MatGo?",
      createAccount: "Fungua Akaunti",
    }
  };
  const currentContent = content[language];

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

    console.log('Login attempt started');
    console.log('Username:', username);
    console.log('Password length:', password.length);

    if (!username || !password) {
      const errorMsg = !username ? 'Username is required' : 'Password is
required';
      console.error('Validation error:', errorMsg);
      toast({
        variant: "destructive",
        title: currentContent.loginFailed,
        description: errorMsg,
      });
      return;
    }

    try {
      // Clear any existing auth data
      localStorage.removeItem('matgoUser');
      localStorage.removeItem('matgoToken');

      // Determine if input is email or phone
      const isEmail = username.includes('@');
      const loginPayload = isEmail
        ? { email: username, password }
        : { phone: username, password };

      console.log('Sending login request to backend...', loginPayload);
      const response = await axios.post('http://localhost:5000/api/auth/login',
        loginPayload,
        {
          headers: {
            'Content-Type': 'application/json',
            'Accept': 'application/json',
            'Origin': 'http://localhost:9002'
          },
          withCredentials: true
        }
      );

      console.log('Login response received:', {
        status: response.status,
        statusText: response.statusText,
        headers: response.headers,
        data: response.data
      });

      if (!response.data || !response.data.user || !response.data.token) {
        console.error('Invalid response format from server:', response.data);
        throw new Error('Invalid response from server');
      }

      const { data } = response;

      // Store user data and token
```

```
      localStorage.setItem('matgoUser', JSON.stringify(data.user));
      localStorage.setItem('matgoToken', data.token);

      toast({
        title: currentContent.loginSuccess,
        description: currentContent.loginRedirect,
        className: "bg-green-500 text-white",
      });

      // Redirect based on user role
      const dashboardPaths = {
        'admin': '/dashboard/admin',
        'sacco_admin': '/dashboard/sacco',
        'driver': '/dashboard/driver',
        'conductor': '/dashboard/conductor',
        'passenger': '/dashboard/passenger'
      } as const;

      type RoleKey = keyof typeof dashboardPaths;
      const rawRole: string = data.user.role === "system_admin" ? "admin" :
data.user.role;
      const role: RoleKey = (["admin", "sacco_admin", "driver", "conductor",
"passenger"].includes(rawRole) ? rawRole : "passenger") as RoleKey;

      // Default to passenger dashboard if role not found
      const targetPath = dashboardPaths[role];
      window.location.href = targetPath;
    } catch (error: any) {
      console.error('Login error:', error);
      console.error('Error response data:', error.response?.data);
      console.error('Error status:', error.response?.status);
      console.error('Error headers:', error.response?.headers);

      let errorMessage = currentContent.invalidCredentials;

      if (error.response) {
        // The request was made and the server responded with a status code
        // that falls out of the range of 2xx
        console.error('Error response data:', error.response.data);
        console.error('Error status:', error.response.status);
        console.error('Error headers:', error.response.headers);

        // Provide more specific error messages based on the response
        if (error.response.status === 401) {
          // Handle 401 Unauthorized specifically
          if (error.response.data) {
            // Check for different possible error message formats
            if (typeof error.response.data === 'string') {
              errorMessage = error.response.data;
            } else if (error.response.data.message) {
              errorMessage = error.response.data.message;
            } else if (error.response.data.error) {
              errorMessage = error.response.data.error;
            } else {
              // If we get an object but can't find a message, stringify it
              try {
                errorMessage = JSON.stringify(error.response.data);
              } catch (e) {
                errorMessage = 'Invalid credentials. Please check your username
and password.';
              }
            }
          } else {
            errorMessage = 'Invalid credentials. Please check your username and
```

```
password.';
          }
        } else if (error.response.status === 403) {
          errorMessage = 'Your account is pending approval. Please contact
support.';
        } else if (error.response.status >= 500) {
          errorMessage = 'Server error. Please try again later.';
        }
      } else if (error.request) {
        // The request was made but no response was received
        console.error('No response received:', error.request);
        errorMessage = 'No response from server. Please check your connection.';
      } else {
        // Something happened in setting up the request that triggered an Error
        console.error('Error setting up request:', error.message);
        errorMessage = `Error: ${error.message}`;
      }

      toast({
        variant: "destructive",
        title: currentContent.loginFailed,
        description: errorMessage,
      });
    }
  };

  return (
    <div className="flex min-h-screen flex-col bg-gradient-to-br from-background
via-muted/30 to-background dark:from-gray-900 dark:via-gray-800/50 dark:to-
gray-900">
      <Header />
      <main className="flex flex-1 items-center justify-center p-4 md:p-6">
        <Card className="w-full max-w-md shadow-2xl animate-fade-in rounded-xl
glassy-card">
          <CardHeader className="text-center space-y-3 pt-8">
            <MatGoIcon className="mx-auto h-20 w-20 text-primary nganya-
flash" />
            <CardTitle className="font-headline text-4xl text-
primary">{currentContent.welcomeBack}</CardTitle>
            <CardDescription className="text-muted-foreground text-
base">{currentContent.loginToAccount}</CardDescription>
          </CardHeader>
          <CardContent className="pt-6 pb-8 px-6 md:px-8">
            <form onSubmit={handleSubmit} className="space-y-6">
              <div className="space-y-2">
                <Label htmlFor="username" className="text-base font-
semibold">{currentContent.phoneEmailLabel}</Label>
                <Input
                  id="username"
                  type="text"
                  placeholder={currentContent.phoneEmailPlaceholder}
                  required
                  className="text-base py-3 px-4 rounded-lg border-2
focus:border-primary focus:ring-primary"
                  value={username}
                  onChange={(e) => setUsername(e.target.value)}
                />
              </div>
              <div className="space-y-2">
                <div className="flex items-center justify-between">
                  <Label htmlFor="password" className="text-base font-
semibold">{currentContent.passwordLabel}</Label>
                  <Link href="/forgot-password" className="text-sm text-primary
hover:underline font-medium">
```

```
                  {currentContent.forgotPassword}
                </Link>
              </div>
              <Input
                id="password"
                type="password"
                required
                placeholder={currentContent.passwordPlaceholder}
                className="text-base py-3 px-4 rounded-lg border-2
focus:border-primary focus:ring-primary"
                value={password}
                onChange={(e) => setPassword(e.target.value)}
              />
            </div>
            <Button type="submit" className="w-full font-bold text-lg py-6
rounded-lg bg-primary text-primary-foreground hover:bg-primary/90 transition-all
duration-300 ease-in-out transform hover:scale-105 btn-glow-primary">
              {currentContent.loginButton} <ArrowRight className="ml-2 h-5
w-5"/>
            </Button>
          </form>
        </CardContent>
        <CardFooter className="flex flex-col items-center gap-3 pb-8">
          <p className="text-sm text-muted-foreground">
            {currentContent.newToMatGo}{" "}
            <Button variant="link" asChild className="text-primary p-0 h-auto
font-semibold text-base hover:underline">
              <Link href="/signup">{currentContent.createAccount}</Link>
            </Button>
          </p>
        </CardFooter>
      </Card>
    </main>
  </div>
 );
}
//admin page//

"use client";

import { useEffect } from "react";
import { useRouter } from "next/navigation";
import { Skeleton } from "@/components/ui/skeleton";

export default function AdminRootPage() {
  const router = useRouter();

  useEffect(() => {
    // In a real app, you'd get the role from an auth context.
    // For this simulation, we check localStorage.
    const storedUser = localStorage.getItem('matgoUser');
    let userRole = 'passenger';
    if (storedUser) {
        try {
            userRole = JSON.parse(storedUser).role;
        } catch (e) { /* default to passenger */ }
    }

    // Redirect to the appropriate dashboard based on the role.
    if (userRole === 'admin') {
      router.replace('/dashboard/admin');
    } else if (userRole === 'sacco_admin') {
      router.replace('/dashboard/sacco');
    } else {
```

```
      // If a non-admin somehow lands here, redirect them away.
      router.replace('/login');
    }
  }, [router]);

  // Display a loading skeleton while the redirect is happening.
  return (
    <div className="h-screen w-screen flex items-center justify-center">
      <div className="space-y-4 w-full p-8">
        <Skeleton className="h-12 w-1/3" />
        <Skeleton className="h-32 w-full" />
        <Skeleton className="h-32 w-full" />
      </div>
    </div>
  );
}
```