Contents

SECTION 29: ADMIN DASHBOARD EXTENSIONS (v3.7.0)

> **NEW in v3.7.0**: Extended admin dashboard capabilities for managing all v3.x features.

29.1 Admin Dashboard Navigation Update

```
// apps/admin-dashboard/src/components/layout/sidebar.tsx

const navigationItems = [
  // Existing items...
  { name: 'Dashboard', href: '/dashboard', icon: LayoutDashboard },
  { name: 'Tenants', href: '/tenants', icon: Building2 },
  { name: 'Users', href: '/users', icon: Users },
  { name: 'Administrators', href: '/administrators', icon: Shield },

  // AI & Models Section
  { type: 'separator', label: 'AI & Models' },
  { name: 'Providers', href: '/providers', icon: Cloud },
  { name: 'Models', href: '/models', icon: Brain },
  { name: 'Model Pricing', href: '/models/pricing', icon: DollarSign }, // NEW
  { name: 'Visual AI', href: '/visual-ai', icon: Image },
  { name: 'Thermal States', href: '/thermal-states', icon: Thermometer },

  // Think Tank Section (NEW)
  { type: 'separator', label: 'Think Tank' },
  { name: 'Think Tank Users', href: '/thinktank/users', icon: UserCircle },
  { name: 'Conversations', href: '/thinktank/conversations', icon: MessageSquare },
  { name: 'Domain Modes', href: '/thinktank/domain-modes', icon: Layers },
  { name: 'Model Categories', href: '/thinktank/model-categories', icon: Grid },

  // Operations Section
  { type: 'separator', label: 'Operations' },
  { name: 'Usage & Billing', href: '/billing', icon: Receipt },
  { name: 'Analytics', href: '/analytics', icon: BarChart },
  { name: 'Error Logs', href: '/errors', icon: AlertTriangle },
  { name: 'Audit Logs', href: '/audit', icon: FileText },

  // Settings Section
  { type: 'separator', label: 'Settings' },
  { name: 'Credentials', href: '/credentials', icon: Key },
  { name: 'System Config', href: '/config', icon: Settings },
];
```

29.2 Think Tank User Management Page

```
// apps/admin-dashboard/src/app/(dashboard)/thinktank/users/page.tsx

'use client';

import { useState } from 'react';
import { useQuery } from '@tanstack/react-query';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { DataTable } from '@/components/ui/data-table';
import { Badge } from '@/components/ui/badge';
import { Button } from '@/components/ui/button';
import { Input } from '@/components/ui/input';
import { Search, Download, UserX, Mail } from 'lucide-react';
import { formatDistanceToNow } from 'date-fns';

const columns = [
  {
    accessorKey: 'email',
    header: 'Email',
    cell: ({ row }) => (
      <div>
        <div className="font-medium">{row.original.email}</div>
        <div className="text-xs text-muted-foreground">
          ID: {row.original.id.slice(0, 8)}...
        </div>
      </div>
    ),
  },
  {
    accessorKey: 'display_name',
    header: 'Display Name',
  },
  {
    accessorKey: 'language',
    header: 'Language',
    cell: ({ row }) => (
      <Badge variant="outline">{row.original.language?.toUpperCase() || 'EN'}</Badge>
    ),
  },
  {
    accessorKey: 'subscription_tier',
    header: 'Tier',
    cell: ({ row }) => {
      const tier = row.original.subscription_tier;
      const colors = {
        free: 'bg-gray-100',
        pro: 'bg-blue-100 text-blue-800',
        team: 'bg-purple-100 text-purple-800',
        enterprise: 'bg-amber-100 text-amber-800',
      };
      return <Badge className={colors[tier] || colors.free}>{tier}</Badge>;
    },
  },
  {
    accessorKey: 'conversation_count',
    header: 'Conversations',
    cell: ({ row }) => row.original.conversation_count?.toLocaleString() || '0',
  },
  {
    accessorKey: 'total_tokens_used',
    header: 'Tokens Used',
    cell: ({ row }) => (row.original.total_tokens_used || 0).toLocaleString(),
  },
  {
    accessorKey: 'total_spent',
    header: 'Total Spent',
    cell: ({ row }) => `$${(row.original.total_spent || 0).toFixed(2)}`,
  },
  {
    accessorKey: 'last_active_at',
    header: 'Last Active',
    cell: ({ row }) =>
      row.original.last_active_at
        ? formatDistanceToNow(new Date(row.original.last_active_at), { addSuffix: true })
        : 'Never',
  },
  {
    accessorKey: 'status',
    header: 'Status',
    cell: ({ row }) => (
      <Badge variant={row.original.status === 'active' ? 'default' : 'destructive'}>
        {row.original.status}
      </Badge>
    ),
  },
];

export default function ThinkTankUsersPage() {
  const [search, setSearch] = useState('');

  const { data: users, isLoading } = useQuery({
    queryKey: ['thinktank-users', search],
    queryFn: () => fetch(`/api/admin/thinktank/users?search=${search}`).then(r => r.json()),
  });

  const { data: stats } = useQuery({
    queryKey: ['thinktank-user-stats'],
    queryFn: () => fetch('/api/admin/thinktank/users/stats').then(r => r.json()),
  });

  return (
    <div className="space-y-6">
      <div className="flex justify-between items-center">
        <div>
          <h1 className="text-2xl font-bold">Think Tank Users</h1>
          <p className="text-muted-foreground">
            Manage Think Tank consumer users
          </p>
        </div>
        <Button variant="outline">
          <Download className="h-4 w-4 mr-2" />
          Export
        </Button>
      </div>

      {/* Stats Cards */}
      <div className="grid gap-4 md:grid-cols-4">
        <Card>
          <CardHeader className="pb-2">
            <CardTitle className="text-sm font-medium text-muted-foreground">
              Total Users
            </CardTitle>
          </CardHeader>
          <CardContent>
            <div className="text-2xl font-bold">{stats?.totalUsers?.toLocaleString() || 0}</di
          </CardContent>
        </Card>
        <Card>
          <CardHeader className="pb-2">
            <CardTitle className="text-sm font-medium text-muted-foreground">
              Active (7d)
      </CardTitle>
    </CardHeader>
    <CardContent>
      <div className="text-2xl font-bold">{stats?.activeUsers7d?.toLocaleString() || 0}<
    </CardContent>
  </Card>
  <Card>
    <CardHeader className="pb-2">
      <CardTitle className="text-sm font-medium text-muted-foreground">
        Pro+ Subscribers
      </CardTitle>
    </CardHeader>
    <CardContent>
      <div className="text-2xl font-bold">{stats?.paidUsers?.toLocaleString() || 0}</div
    </CardContent>
  </Card>
  <Card>
    <CardHeader className="pb-2">
      <CardTitle className="text-sm font-medium text-muted-foreground">
        Total Revenue
      </CardTitle>
    </CardHeader>
    <CardContent>
      <div className="text-2xl font-bold">${stats?.totalRevenue?.toLocaleString() || 0}<
    </CardContent>
  </Card>
</div>

{/* Search & Table */}
<Card>
  <CardHeader>
    <div className="flex items-center gap-4">
      <div className="relative flex-1 max-w-sm">
        <Search className="absolute left-3 top-1/2 -translate-y-1/2 h-4 w-4 text-muted-f
        <Input
          placeholder="Search by email or name..."
          value={search}
          onChange={(e) => setSearch(e.target.value)}
          className="pl-9"
        />
      </div>
    </div>
  </CardHeader>
  <CardContent>
    <DataTable
      columns={columns}
      data={users?.data || []}
      isLoading={isLoading}
```

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

29.3 Domain Modes Configuration Page

```
// apps/admin-dashboard/src/app/(dashboard)/thinktank/domain-modes/page.tsx

'use client';

import { useState } from 'react';
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query';
import { Card, CardContent, CardHeader, CardTitle, CardDescription } from '@/components/ui/car
import { Switch } from '@/components/ui/switch';
import { Button } from '@/components/ui/button';
import { Input } from '@/components/ui/input';
import { Label } from '@/components/ui/label';
import { Textarea } from '@/components/ui/textarea';
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from '@/components/ui/
import { Badge } from '@/components/ui/badge';
import { toast } from 'sonner';
import {
  Stethoscope, Scale, Code2, Lightbulb, GraduationCap,
  PenTool, FlaskConical, Save
} from 'lucide-react';

const DOMAIN_MODES = [
  { id: 'general', name: 'General', icon: Lightbulb, description: 'Default mode for general qu
  { id: 'medical', name: 'Medical', icon: Stethoscope, description: 'Healthcare and medical to
  { id: 'legal', name: 'Legal', icon: Scale, description: 'Legal research and analysis' },
  { id: 'code', name: 'Code', icon: Code2, description: 'Programming and development' },
  { id: 'academic', name: 'Academic', icon: GraduationCap, description: 'Research and educatio
  { id: 'creative', name: 'Creative', icon: PenTool, description: 'Writing and content creatio
  { id: 'scientific', name: 'Scientific', icon: FlaskConical, description: 'Scientific researc
];

export default function DomainModesPage() {
  const queryClient = useQueryClient();

  const { data: config } = useQuery({
    queryKey: ['domain-modes-config'],
    queryFn: () => fetch('/api/admin/thinktank/domain-modes').then(r => r.json()),
  });
```

```
const { data: models } = useQuery({
  queryKey: ['available-models'],
  queryFn: () => fetch('/api/admin/models?enabled=true').then(r => r.json()),
});

const updateMutation = useMutation({
  mutationFn: (data: any) =>
    fetch('/api/admin/thinktank/domain-modes', {
      method: 'PUT',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(data),
    }).then(r => r.json()),
  onSuccess: () => {
    queryClient.invalidateQueries({ queryKey: ['domain-modes-config'] });
    toast.success('Domain mode configuration saved');
  },
});

const [localConfig, setLocalConfig] = useState(config || {});

return (
  <div className="space-y-6">
    <div className="flex justify-between items-center">
      <div>
        <h1 className="text-2xl font-bold">Domain Modes</h1>
        <p className="text-muted-foreground">
          Configure specialized AI modes for different use cases
        </p>
      </div>
      <Button onClick={() => updateMutation.mutate(localConfig)}>
        <Save className="h-4 w-4 mr-2" />
        Save Changes
      </Button>
    </div>

    <div className="grid gap-4">
      {DOMAIN_MODES.map((mode) => {
        const ModeIcon = mode.icon;
        const modeConfig = localConfig?.modes?.[mode.id] || {};

        return (
          <Card key={mode.id}>
            <CardHeader>
              <div className="flex items-center justify-between">
                <div className="flex items-center gap-3">
                  <div className="p-2 rounded-lg bg-primary/10">
                    <ModeIcon className="h-5 w-5 text-primary" />
      </div>
      <div>
        <CardTitle className="text-lg">{mode.name}</CardTitle>
        <CardDescription>{mode.description}</CardDescription>
      </div>
    </div>
    <Switch
      checked={modeConfig.enabled !== false}
      onCheckedChange={(checked) =>
        setLocalConfig({
          ...localConfig,
          modes: {
            ...localConfig.modes,
            [mode.id]: { ...modeConfig, enabled: checked },
          },
        })
      }
    />
  </div>
</CardHeader>
<CardContent className="space-y-4">
  <div className="grid gap-4 md:grid-cols-2">
    <div className="space-y-2">
      <Label>Default Model</Label>
      <Select
        value={modeConfig.defaultModel || 'auto'}
        onValueChange={(value) =>
          setLocalConfig({
            ...localConfig,
            modes: {
              ...localConfig.modes,
              [mode.id]: { ...modeConfig, defaultModel: value },
            },
          })
        }
      >
        <SelectTrigger>
          <SelectValue placeholder="Auto (RADIANT Brain)" />
        </SelectTrigger>
        <SelectContent>
          <SelectItem value="auto">Auto (RADIANT Brain)</SelectItem>
          {(models?.data || []).map((model: any) => (
            <SelectItem key={model.id} value={model.id}>
              {model.display_name}
            </SelectItem>
          ))}
        </SelectContent>
      </Select>
```

```
         </div>

         <div className="space-y-2">
           <Label>Temperature</Label>
           <Input
             type="number"
             min="0"
             max="2"
             step="0.1"
             value={modeConfig.temperature || 0.7}
             onChange={(e) =>
               setLocalConfig({
                 ...localConfig,
                 modes: {
                   ...localConfig.modes,
                   [mode.id]: { ...modeConfig, temperature: parseFloat(e.target.value
                 },
               })
             }
           />
         </div>
        </div>

        <div className="space-y-2">
         <Label>System Prompt Override</Label>
         <Textarea
           placeholder="Optional: Custom system prompt for this mode..."
           value={modeConfig.systemPrompt || ''}
           onChange={(e) =>
             setLocalConfig({
               ...localConfig,
               modes: {
                 ...localConfig.modes,
                 [mode.id]: { ...modeConfig, systemPrompt: e.target.value },
               },
             })
           }
           rows={3}
         />
        </div>
       </CardContent>
      </Card>
     );
    })}
   </div>
  </div>
 );
}
```