****Dashboard****

```
import React, { useState, useEffect } from "react";
import { Agent, Task } from "@/entities/all";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/
card";
import { Badge } from "@/components/ui/badge";
import { Button } from "@/components/ui/button";
import { Progress } from "@/components/ui/progress";
import {
  Brain,
  Zap,
  Clock,
  TrendingUp,
  Play,
  Pause,
  Square,
  Users,
  Activity,
  BarChart3
} from "lucide-react";
import { motion, AnimatePresence } from "framer-motion";

import SystemMetrics from "../components/dashboard/SystemMetrics";
import AgentStatusGrid from "../components/dashboard/AgentStatusGrid";
import TaskQueue from "../components/dashboard/TaskQueue";
import FrameworkSelector from "../components/dashboard/FrameworkSelector";

export default function Dashboard() {
  const [agents, setAgents] = useState([]);
  const [tasks, setTasks] = useState([]);
  const [systemMetrics, setSystemMetrics] = useState({
    totalAgents: 0,
    activeAgents: 0,
    completedTasks: 0,
    avgPerformance: 0
  });
  const [isLoading, setIsLoading] = useState(true);

  useEffect(() => {
    loadDashboardData();
    const interval = setInterval(loadDashboardData, 5000); // Real-time updates
```

```
    return () => clearInterval(interval);
  }, []);

  const loadDashboardData = async () => {
    try {
      const [agentsData, tasksData] = await Promise.all([
        Agent.list('-last_activity', 20),
        Task.list('-created_date', 10)
      ]);

      setAgents(agentsData);
      setTasks(tasksData);

      // Calculate metrics
      const activeAgents = agentsData.filter(a => a.status === 'running').length;
      const completedTasks = tasksData.filter(t => t.status === 'completed').length;
      const avgPerformance = agentsData.reduce((sum, a) => sum +
(a.performance_score || 0), 0) / agentsData.length || 0;

      setSystemMetrics({
        totalAgents: agentsData.length,
        activeAgents,
        completedTasks,
        avgPerformance: Math.round(avgPerformance)
      });

      setIsLoading(false);
    } catch (error) {
      console.error('Error loading dashboard data:', error);
      setIsLoading(false);
    }
  };

  return (
    <div className="min-h-screen p-6" style={{background: 'linear-
gradient(135deg, #0a0a0a 0%, #1a1a1a 100%)'}}>
      <div className="max-w-7xl mx-auto space-y-8">
        {/* Header */}
        <motion.div
          initial={{ opacity: 0, y: -20 }}
          animate={{ opacity: 1, y: 0 }}
          className="flex flex-col lg:flex-row justify-between items-start lg:items-
center gap-6"
        >
```

```
          <div>
            <h1 className="text-4xl font-bold bg-gradient-to-r from-white to-
orange-400 bg-clip-text text-transparent">
              Orchestration Hub
            </h1>
            <p className="text-gray-400 mt-2 text-lg">Real-time AI agent monitoring
and control</p>
          </div>
          <div className="flex gap-3">
            <Button className="bg-orange-500 hover:bg-orange-600 text-white
shadow-lg hover:shadow-orange-500/25 transition-all duration-300">
              <Play className="w-4 h-4 mr-2" />
              Deploy Agent
            </Button>
            <Button variant="outline" className="border-orange-500/30 text-
orange-400 hover:bg-orange-500/10">
              <BarChart3 className="w-4 h-4 mr-2" />
              View Analytics
            </Button>
          </div>
        </motion.div>

        {/* System Metrics */}
        <SystemMetrics metrics={systemMetrics} isLoading={isLoading} />

        {/* Main Grid */}
        <div className="grid lg:grid-cols-3 gap-8">
          {/* Agent Status Grid */}
          <div className="lg:col-span-2">
            <AgentStatusGrid agents={agents} isLoading={isLoading} />
          </div>

          {/* Sidebar Content */}
          <div className="space-y-6">
            <TaskQueue tasks={tasks} isLoading={isLoading} />
            <FrameworkSelector />
          </div>
        </div>

        {/* Real-time Activity Feed */}
        <motion.div
          initial={{ opacity: 0 }}
          animate={{ opacity: 1 }}
          transition={{ delay: 0.3 }}
```

```
      >
        <Card className="glass-effect border-orange-500/20">
          <CardHeader>
            <CardTitle className="flex items-center gap-2 text-white">
              <Activity className="w-5 h-5 text-orange-400" />
              Live Activity Stream
            </CardTitle>
          </CardHeader>
          <CardContent>
            <div className="space-y-3 max-h-60 overflow-y-auto">
              {tasks.slice(0, 5).map((task, index) => (
                <motion.div
                  key={task.id}
                  initial={{ opacity: 0, x: -20 }}
                  animate={{ opacity: 1, x: 0 }}
                  transition={{ delay: index * 0.1 }}
                  className="flex items-center justify-between p-3 bg-white/5
rounded-lg border border-white/10"
                >
                  <div className="flex items-center gap-3">
                    <div className={`w-2 h-2 rounded-full ${
                      task.status === 'completed' ? 'bg-green-400' :
                      task.status === 'in_progress' ? 'bg-orange-400' :
                      'bg-gray-400'
                    }`} />
                    <span className="text-sm text-gray-200">{task.title}</span>
                  </div>
                  <Badge variant="outline" className="border-orange-500/30 text-
orange-400">
                    {task.framework}
                  </Badge>
                </motion.div>
              ))}
            </div>
          </CardContent>
        </Card>
      </motion.div>
    </div>
  </div>
 );
}
```

****Agents****

```
import React, { useState, useEffect } from "react";
import { Agent } from "@/entities/all";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/
card";
import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input";
import { Badge } from "@/components/ui/badge";
import {
  Users,
  Plus,
  Search,
  Filter,
  Brain,
  Play,
  Pause,
  Settings,
  Trash2
} from "lucide-react";
import { motion, AnimatePresence } from "framer-motion";

import AgentCard from "../components/agents/AgentCard";
import CreateAgentModal from "../components/agents/CreateAgentModal";
import AgentFilters from "../components/agents/AgentFilters";

export default function Agents() {
  const [agents, setAgents] = useState([]);
  const [filteredAgents, setFilteredAgents] = useState([]);
  const [searchTerm, setSearchTerm] = useState("");
  const [showCreateModal, setShowCreateModal] = useState(false);
  const [selectedFramework, setSelectedFramework] = useState("all");
  const [selectedStatus, setSelectedStatus] = useState("all");
  const [isLoading, setIsLoading] = useState(true);

  useEffect(() => {
    loadAgents();
  }, []);

  useEffect(() => {
    filterAgents();
  }, [agents, searchTerm, selectedFramework, selectedStatus]);

  const loadAgents = async () => {
    try {
```

```
    const data = await Agent.list('-last_activity');
    setAgents(data);
    setIsLoading(false);
  } catch (error) {
    console.error('Error loading agents:', error);
    setIsLoading(false);
  }
};

const filterAgents = () => {
  let filtered = agents;

  if (searchTerm) {
    filtered = filtered.filter(agent =>
      agent.name.toLowerCase().includes(searchTerm.toLowerCase()) ||
      agent.framework.toLowerCase().includes(searchTerm.toLowerCase())
    );
  }

  if (selectedFramework !== "all") {
    filtered = filtered.filter(agent => agent.framework === selectedFramework);
  }

  if (selectedStatus !== "all") {
    filtered = filtered.filter(agent => agent.status === selectedStatus);
  }

  setFilteredAgents(filtered);
};

const handleCreateAgent = async (agentData) => {
  try {
    await Agent.create(agentData);
    loadAgents();
    setShowCreateModal(false);
  } catch (error) {
    console.error('Error creating agent:', error);
  }
};

const handleAgentAction = async (agentId, action) => {
  try {
    const agent = agents.find(a => a.id === agentId);
    if (!agent) return;
```

```
      let newStatus = agent.status;
      if (action === 'start') newStatus = 'running';
      if (action === 'pause') newStatus = 'paused';
      if (action === 'stop') newStatus = 'idle';

      await Agent.update(agentId, {
        status: newStatus,
        last_activity: new Date().toISOString()
      });
      loadAgents();
    } catch (error) {
      console.error('Error updating agent:', error);
    }
  };

  return (
    <div className="min-h-screen p-6" style={{background: 'linear-
gradient(135deg, #0a0a0a 0%, #1a1a1a 100%)'}}>
      <div className="max-w-7xl mx-auto space-y-8">
        {/* Header */}
        <motion.div
          initial={{ opacity: 0, y: -20 }}
          animate={{ opacity: 1, y: 0 }}
          className="flex flex-col lg:flex-row justify-between items-start lg:items-
center gap-6"
        >
          <div>
            <h1 className="text-4xl font-bold bg-gradient-to-r from-white to-
orange-400 bg-clip-text text-transparent">
              Agent Management
            </h1>
            <p className="text-gray-400 mt-2 text-lg">Deploy, monitor, and control
AI agents across frameworks</p>
          </div>
          <Button
            onClick={() => setShowCreateModal(true)}
            className="bg-orange-500 hover:bg-orange-600 text-white shadow-lg
hover:shadow-orange-500/25 transition-all duration-300"
          >
            <Plus className="w-4 h-4 mr-2" />
            Deploy New Agent
          </Button>
        </motion.div>
```

```
        {/* Search and Filters */}
        <motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ delay: 0.1 }}
          className="flex flex-col lg:flex-row gap-4"
        >
          <div className="relative flex-1">
            <Search className="absolute left-3 top-1/2 transform -translate-y-1/2
text-gray-400 w-4 h-4" />
            <Input
              placeholder="Search agents by name or framework..."
              value={searchTerm}
              onChange={(e) => setSearchTerm(e.target.value)}
              className="pl-10 bg-white/5 border-white/10 text-white
placeholder:text-gray-400 focus:border-orange-500/50"
            />
          </div>
          <AgentFilters
            selectedFramework={selectedFramework}
            selectedStatus={selectedStatus}
            onFrameworkChange={setSelectedFramework}
            onStatusChange={setSelectedStatus}
          />
        </motion.div>

        {/* Agent Grid */}
        <motion.div
          initial={{ opacity: 0 }}
          animate={{ opacity: 1 }}
          transition={{ delay: 0.2 }}
        >
          <div className="grid md:grid-cols-2 lg:grid-cols-3 gap-6">
            <AnimatePresence>
              {filteredAgents.map((agent, index) => (
                <motion.div
                  key={agent.id}
                  initial={{ opacity: 0, scale: 0.9 }}
                  animate={{ opacity: 1, scale: 1 }}
                  exit={{ opacity: 0, scale: 0.9 }}
                  transition={{ delay: index * 0.05 }}
                >
                  <AgentCard
```

```
                agent={agent}
                onAction={handleAgentAction}
              />
            </motion.div>
          ))}
        </AnimatePresence>
      </div>

      {filteredAgents.length === 0 && !isLoading && (
        <div className="text-center py-12">
          <Brain className="w-16 h-16 text-gray-500 mx-auto mb-4" />
          <h3 className="text-xl font-semibold text-gray-300 mb-2">No agents
found</h3>
          <p className="text-gray-400 mb-6">
            {searchTerm || selectedFramework !== "all" || selectedStatus !== "all"
              ? "Try adjusting your search or filters"
              : "Deploy your first AI agent to get started"
            }
          </p>
          {!searchTerm && selectedFramework === "all" && selectedStatus ===
"all" && (
            <Button
              onClick={() => setShowCreateModal(true)}
              className="bg-orange-500 hover:bg-orange-600"
            >
              <Plus className="w-4 h-4 mr-2" />
              Deploy First Agent
            </Button>
          )}
        </div>
      )}
    </motion.div>

    {/* Create Agent Modal */}
    <CreateAgentModal
      isOpen={showCreateModal}
      onClose={() => setShowCreateModal(false)}
      onSubmit={handleCreateAgent}
    />
  </div>
 </div>
 );
}
```

****Settings*****

```
import React, { useState, useEffect } from "react";
import { APIKey } from "@/entities/all";
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 { Badge } from "@/components/ui/badge";
import {
  Settings as SettingsIcon, // Renamed to avoid conflict with the component name
  Key,
  Plus,
  Eye,
  EyeOff,
  Trash2,
  CheckCircle,
  AlertCircle,
  Brain
} from "lucide-react";
import { motion, AnimatePresence } from "framer-motion";
import {
  Dialog,
  DialogContent,
  DialogHeader,
  DialogTitle,
} from "@/components/ui/dialog";
import {
  Select,
  SelectContent,
  SelectItem,
  SelectTrigger,
  SelectValue,
} from "@/components/ui/select";

const providers = [
  { id: "cerebras", name: "Cerebras", color: "bg-orange-500" },
  { id: "openai", name: "OpenAI", color: "bg-green-500" },
  { id: "anthropic", name: "Anthropic", color: "bg-purple-500" },
  { id: "google", name: "Google AI", color: "bg-blue-500" },
  { id: "cohere", name: "Cohere", color: "bg-pink-500" },
```

```
  { id: "custom", name: "Custom", color: "bg-gray-500" },
];

export default function Settings() {
  const [apiKeys, setApiKeys] = useState([]);
  const [showCreateModal, setShowCreateModal] = useState(false);
  const [visibleKeys, setVisibleKeys] = useState(new Set());
  const [isLoading, setIsLoading] = useState(true);

  useEffect(() => {
    loadAPIKeys();
  }, []);

  const loadAPIKeys = async () => {
    try {
      const data = await APIKey.list('-created_date');
      setApiKeys(data);
      setIsLoading(false);
    } catch (error) {
      console.error('Error loading API keys:', error);
      setIsLoading(false);
    }
  };

  const handleCreateAPIKey = async (keyData) => {
    try {
      await APIKey.create({
        ...keyData,
        usage_count: 0,
        last_used: null
      });
      loadAPIKeys();
      setShowCreateModal(false);
    } catch (error) {
      console.error('Error creating API key:', error);
    }
  };

  const handleDeleteAPIKey = async (keyId) => {
    try {
      await APIKey.delete(keyId);
      loadAPIKeys();
    } catch (error) {
      console.error('Error deleting API key:', error);
```

```
  }
 };

 const toggleKeyVisibility = (keyId) => {
  const newVisible = new Set(visibleKeys);
  if (newVisible.has(keyId)) {
   newVisible.delete(keyId);
  } else {
   newVisible.add(keyId);
  }
  setVisibleKeys(newVisible);
 };

 const maskKey = (key) => {
  if (!key) return '';
  return key.substring(0, 8) + '••••••••••••••••' + key.substring(key.length - 4);
 };

 const getStatusColor = (status) => {
  switch (status) {
   case 'active': return 'bg-green-500/20 text-green-400 border-green-500/30';
   case 'inactive': return 'bg-gray-500/20 text-gray-400 border-gray-500/30';
   case 'expired': return 'bg-red-500/20 text-red-400 border-red-500/30';
   case 'invalid': return 'bg-orange-500/20 text-orange-400 border-
orange-500/30';
   default: return 'bg-gray-500/20 text-gray-400 border-gray-500/30';
  }
 };

 const getProviderInfo = (providerId) => {
  return providers.find(p => p.id === providerId) || providers[providers.length - 1];
 };

 return (
  <div className="min-h-screen p-6" style={{background: 'linear-
gradient(135deg, #0a0a0a 0%, #1a1a1a 100%)'}}>
   <div className="max-w-6xl mx-auto space-y-8">
    {/* Header */}
    <motion.div
     initial={{ opacity: 0, y: -20 }}
     animate={{ opacity: 1, y: 0 }}
     className="flex flex-col lg:flex-row justify-between items-start lg:items-
center gap-6"
    >
```

```
        <div>
          <h1 className="text-4xl font-bold bg-gradient-to-r from-white to-
orange-400 bg-clip-text text-transparent">
            API Configuration
          </h1>
          <p className="text-gray-400 mt-2 text-lg">Manage API keys and system
configuration</p>
        </div>
        <Button
          onClick={() => setShowCreateModal(true)}
          className="bg-orange-500 hover:bg-orange-600 text-white shadow-lg
hover:shadow-orange-500/25 transition-all duration-300"
        >
          <Plus className="w-4 h-4 mr-2" />
          Add API Key
        </Button>
      </motion.div>

      {/* API Keys Grid */}
      <motion.div
        initial={{ opacity: 0, y: 20 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ delay: 0.1 }}
      >
        <Card className="glass-effect border-white/10">
          <CardHeader>
            <CardTitle className="flex items-center gap-2 text-white">
              <Key className="w-5 h-5 text-orange-400" />
              API Keys Management
              <Badge variant="outline" className="ml-auto border-orange-500/30
text-orange-400">
                {apiKeys.filter(k => k.status === 'active').length} Active
              </Badge>
            </CardTitle>
          </CardHeader>
          <CardContent>
            <div className="space-y-4">
              <AnimatePresence>
                {apiKeys.map((apiKey, index) => {
                  const provider = getProviderInfo(apiKey.provider);
                  return (
                    <motion.div
                      key={apiKey.id}
                      initial={{ opacity: 0, x: -20 }}
```

```
                animate={{ opacity: 1, x: 0 }}
                exit={{ opacity: 0, x: 20 }}
                transition={{ delay: index * 0.05 }}
                className="p-4 bg-white/5 rounded-lg border border-white/10
hover:border-orange-500/20 transition-all duration-300"
              >
                <div className="flex items-center justify-between mb-3">
                  <div className="flex items-center gap-3">
                    <div className={`w-10 h-10 ${provider.color} rounded-lg flex
items-center justify-center`}>
                      <Brain className="w-5 h-5 text-white" />
                    </div>
                    <div>
                      <h3 className="font-semibold text-white">{apiKey.name}</
h3>
                      <div className="flex items-center gap-2 mt-1">
                        <Badge variant="outline" className="border-white/20 text-
gray-300">
                          {provider.name}
                        </Badge>
                        <Badge variant="outline"
className={getStatusColor(apiKey.status)}>
                          {apiKey.status === 'active' && <CheckCircle className="w-3
h-3 mr-1" />}
                          {apiKey.status === 'invalid' && <AlertCircle className="w-3
h-3 mr-1" />}
                          {apiKey.status}
                        </Badge>
                      </div>
                    </div>
                  </div>
                  <div className="flex items-center gap-2">
                    <Button
                      variant="ghost"
                      size="icon"
                      onClick={() => toggleKeyVisibility(apiKey.id)}
                      className="text-gray-400 hover:text-white"
                    >
                      {visibleKeys.has(apiKey.id) ?
                        <EyeOff className="w-4 h-4" /> :
                        <Eye className="w-4 h-4" />
                      }
                    </Button>
                    <Button
```

```
                    variant="ghost"
                    size="icon"
                    onClick={() => handleDeleteAPIKey(apiKey.id)}
                    className="text-red-400 hover:text-red-300 hover:bg-
red-500/10"
                  >
                    <Trash2 className="w-4 h-4" />
                  </Button>
                </div>
              </div>

              <div className="space-y-2">
                <div className="flex items-center gap-2">
                  <span className="text-sm text-gray-400 min-w-[60px]">Key:</
span>
                  <code className="text-sm text-white bg-black/30 px-2 py-1
rounded font-mono flex-1">
                    {visibleKeys.has(apiKey.id) ? apiKey.key_value :
maskKey(apiKey.key_value)}
                  </code>
                </div>

                {(apiKey.usage_count > 0 || apiKey.last_used) && (
                  <div className="grid grid-cols-2 gap-4 text-sm">
                    <div>
                      <span className="text-gray-400">Usage Count:</span>
                      <span className="text-white ml-2 font-
medium">{apiKey.usage_count || 0}</span>
                    </div>
                    {apiKey.last_used && (
                      <div>
                        <span className="text-gray-400">Last Used:</span>
                        <span className="text-white ml-2 font-medium">
                          {new Date(apiKey.last_used).toLocaleDateString()}
                        </span>
                      </div>
                    )}
                  </div>
                )}
              </div>
            </motion.div>
          );
        })}
      </AnimatePresence>
```

```
            {apiKeys.length === 0 && !isLoading && (
              <div className="text-center py-8">
                <Key className="w-12 h-12 text-gray-500 mx-auto mb-4" />
                <h3 className="text-lg font-semibold text-gray-300 mb-2">No API
Keys Configured</h3>
                <p className="text-gray-400 mb-4">Add your first API key to start
using AI frameworks</p>
                <Button
                  onClick={() => setShowCreateModal(true)}
                  className="bg-orange-500 hover:bg-orange-600"
                >
                  <Plus className="w-4 h-4 mr-2" />
                  Add First API Key
                </Button>
              </div>
            )}
          </div>
        </CardContent>
      </Card>
    </motion.div>

    {/* System Configuration */}
    <motion.div
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ delay: 0.2 }}
    >
      <Card className="glass-effect border-white/10">
        <CardHeader>
          <CardTitle className="flex items-center gap-2 text-white">
            <SettingsIcon className="w-5 h-5 text-orange-400" /> {/* Changed to
SettingsIcon */}
            System Configuration
          </CardTitle>
        </CardHeader>
        <CardContent>
          <div className="grid md:grid-cols-2 gap-6">
            <div className="space-y-4">
              <div>
                <Label className="text-white font-medium">Default Framework</
Label>
                <Select defaultValue="cerebras">
                  <SelectTrigger className="mt-2 bg-white/5 border-white/10 text-
```

```
white">
                  <SelectValue />
                </SelectTrigger>
                <SelectContent>
                  <SelectItem value="cerebras">Cerebras</SelectItem>
                  <SelectItem value="autogen">AutoGen</SelectItem>
                  <SelectItem value="crewai">CrewAI</SelectItem>
                  <SelectItem value="langgraph">LangGraph</SelectItem>
                </SelectContent>
              </Select>
            </div>
            <div>
              <Label className="text-white font-medium">Max Concurrent
Agents</Label>
              <Input
                type="number"
                defaultValue="10"
                className="mt-2 bg-white/5 border-white/10 text-white"
              />
            </div>
          </div>
          <div className="space-y-4">
            <div>
              <Label className="text-white font-medium">Log Level</Label>
              <Select defaultValue="info">
                <SelectTrigger className="mt-2 bg-white/5 border-white/10 text-
white">
                  <SelectValue />
                </SelectTrigger>
                <SelectContent>
                  <SelectItem value="debug">Debug</SelectItem>
                  <SelectItem value="info">Info</SelectItem>
                  <SelectItem value="warning">Warning</SelectItem>
                  <SelectItem value="error">Error</SelectItem>
                </SelectContent>
              </Select>
            </div>
            <div>
              <Label className="text-white font-medium">Auto-save Interval
(minutes)</Label>
              <Input
                type="number"
                defaultValue="5"
                className="mt-2 bg-white/5 border-white/10 text-white"
```

```
                />
              </div>
            </div>
          </div>
          <div className="flex justify-end mt-6">
            <Button className="bg-orange-500 hover:bg-orange-600">
              Save Configuration
            </Button>
          </div>
        </CardContent>
      </Card>
    </motion.div>

    {/* Create API Key Modal */}
    <CreateAPIKeyModal
      isOpen={showCreateModal}
      onClose={() => setShowCreateModal(false)}
      onSubmit={handleCreateAPIKey}
      providers={providers}
    />
  </div>
 </div>
);
}

function CreateAPIKeyModal({ isOpen, onClose, onSubmit, providers }) {
 const [formData, setFormData] = useState({
  name: "",
  provider: "",
  key_value: "",
  rate_limit: ""
 });

 const handleSubmit = (e) => {
  e.preventDefault();
  if (!formData.name || !formData.provider || !formData.key_value) return;

  onSubmit({
   ...formData,
   status: "active",
   rate_limit: formData.rate_limit ? parseInt(formData.rate_limit) : null
  });
```

```
  setFormData({ name: "", provider: "", key_value: "", rate_limit: "" });
 };

 return (
  <Dialog open={isOpen} onOpenChange={onClose}>
   <DialogContent className="sm:max-w-lg glass-effect border-white/20">
    <DialogHeader>
     <DialogTitle className="flex items-center gap-2 text-white text-xl">
      <Key className="w-6 h-6 text-orange-400" />
      Add New API Key
     </DialogTitle>
    </DialogHeader>

    <form onSubmit={handleSubmit} className="space-y-4 mt-6">
     <div>
      <Label className="text-white font-medium">Key Name</Label>
      <Input
       placeholder="e.g., Cerebras Production Key"
       value={formData.name}
       onChange={(e) => setFormData({ ...formData, name: e.target.value })}
       className="mt-2 bg-white/5 border-white/10 text-white
placeholder:text-gray-400"
       required
      />
     </div>

     <div>
      <Label className="text-white font-medium">Provider</Label>
      <Select value={formData.provider} onValueChange={(value) =>
setFormData({ ...formData, provider: value })}>
       <SelectTrigger className="mt-2 bg-white/5 border-white/10 text-
white">
        <SelectValue placeholder="Select API provider" />
       </SelectTrigger>
       <SelectContent>
        {providers.map((provider) => (
         <SelectItem key={provider.id} value={provider.id}>
          {provider.name}
         </SelectItem>
        ))}
       </SelectContent>
      </Select>
     </div>
```

```
        <div>
          <Label className="text-white font-medium">API Key</Label>
          <Input
            type="password"
            placeholder="Paste your API key here"
            value={formData.key_value}
            onChange={(e) => setFormData({ ...formData, key_value: e.target.value })}
            className="mt-2 bg-white/5 border-white/10 text-white
placeholder:text-gray-400"
            required
          />
        </div>

        <div>
          <Label className="text-white font-medium">Rate Limit (requests/
minute)</Label>
          <Input
            type="number"
            placeholder="Optional rate limit"
            value={formData.rate_limit}
            onChange={(e) => setFormData({ ...formData, rate_limit: e.target.value })}
            className="mt-2 bg-white/5 border-white/10 text-white
placeholder:text-gray-400"
          />
        </div>

        <div className="flex justify-end gap-3 pt-4">
          <Button type="button" variant="outline" onClick={onClose}
className="border-white/20 text-gray-300">
            Cancel
          </Button>
          <Button type="submit" className="bg-orange-500 hover:bg-
orange-600">
            Add API Key
          </Button>
        </div>
      </form>
    </DialogContent>
  </Dialog>
 );
}
```

****System Metrics****

```
import React from "react";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/
card";
import { Brain, Zap, Clock, TrendingUp } from "lucide-react";
import { motion } from "framer-motion";
import { Skeleton } from "@/components/ui/skeleton";

const MetricCard = ({ title, value, icon: Icon, gradient, delay, isLoading }) => (
  <motion.div
    initial={{ opacity: 0, y: 20 }}
    animate={{ opacity: 1, y: 0 }}
    transition={{ delay }}
  >
    <Card className="glass-effect border-white/10 hover:border-orange-500/30
transition-all duration-300">
      <CardContent className="p-6">
        <div className="flex items-center justify-between">
          <div>
            <p className="text-sm font-medium text-gray-400 uppercase tracking-
wider">{title}</p>
            {isLoading ? (
              <Skeleton className="h-8 w-16 mt-2" />
            ) : (
              <p className="text-3xl font-bold text-white mt-2">{value}</p>
            )}
          </div>
          <div className={`p-4 rounded-xl ${gradient} bg-opacity-20 border border-
white/10`}>
            <Icon className="w-6 h-6 text-white" />
          </div>
        </div>
      </CardContent>
    </Card>
  </motion.div>
);

export default function SystemMetrics({ metrics, isLoading }) {
  const metricCards = [
    {
      title: "Total Agents",
      value: metrics.totalAgents,
```

```
      icon: Brain,
      gradient: "bg-gradient-to-r from-blue-500 to-purple-600",
      delay: 0.1
    },
    {
      title: "Active Agents",
      value: metrics.activeAgents,
      icon: Zap,
      gradient: "bg-gradient-to-r from-orange-500 to-red-600",
      delay: 0.2
    },
    {
      title: "Completed Tasks",
      value: metrics.completedTasks,
      icon: Clock,
      gradient: "bg-gradient-to-r from-green-500 to-emerald-600",
      delay: 0.3
    },
    {
      title: "Avg Performance",
      value: `${metrics.avgPerformance}%`,
      icon: TrendingUp,
      gradient: "bg-gradient-to-r from-purple-500 to-pink-600",
      delay: 0.4
    }
  ];

  return (
    <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6">
      {metricCards.map((metric, index) => (
        <MetricCard
          key={metric.title}
          {...metric}
          isLoading={isLoading}
        />
      ))}
    </div>
  );
}
```

*****AgentStatusGrid*****

```
import React from "react";
```

```
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/
card";
import { Badge } from "@/components/ui/badge";
import { Button } from "@/components/ui/button";
import { Progress } from "@/components/ui/progress";
import { Users, Play, Pause, Square, MoreHorizontal } from "lucide-react";
import { motion, AnimatePresence } from "framer-motion";
import { Skeleton } from "@/components/ui/skeleton";
import {
  DropdownMenu,
  DropdownMenuContent,
  DropdownMenuItem,
  DropdownMenuTrigger,
} from "@/components/ui/dropdown-menu";

const frameworkColors = {
  autogen: "bg-blue-500/20 text-blue-400 border-blue-500/30",
  crewai: "bg-purple-500/20 text-purple-400 border-purple-500/30",
  langgraph: "bg-green-500/20 text-green-400 border-green-500/30",
  cerebras: "bg-orange-500/20 text-orange-400 border-orange-500/30",
  default: "bg-gray-500/20 text-gray-400 border-gray-500/30"
};

const statusColors = {
  running: "bg-green-400",
  idle: "bg-gray-400",
  paused: "bg-yellow-400",
  error: "bg-red-400",
  completed: "bg-blue-400"
};

export default function AgentStatusGrid({ agents, isLoading }) {
  if (isLoading) {
    return (
      <Card className="glass-effect border-white/10">
        <CardHeader>
          <CardTitle className="flex items-center gap-2 text-white">
            <Users className="w-5 h-5 text-orange-400" />
            Agent Status Grid
          </CardTitle>
        </CardHeader>
        <CardContent>
          <div className="grid md:grid-cols-2 gap-4">
            {Array(4).fill(0).map((_, i) => (
```

```
          <div key={i} className="p-4 bg-white/5 rounded-lg border border-white/
10">
            <Skeleton className="h-4 w-32 mb-2" />
            <Skeleton className="h-3 w-20 mb-3" />
            <Skeleton className="h-2 w-full mb-2" />
            <div className="flex gap-2">
              <Skeleton className="h-8 w-16" />
              <Skeleton className="h-8 w-16" />
            </div>
          </div>
        ))}
      </div>
    </CardContent>
  </Card>
 );
}

 return (
  <Card className="glass-effect border-white/10">
    <CardHeader>
      <CardTitle className="flex items-center gap-2 text-white">
        <Users className="w-5 h-5 text-orange-400" />
        Agent Status Grid
        <Badge variant="outline" className="ml-auto border-orange-500/30 text-
orange-400">
          {agents.length} Active
        </Badge>
      </CardTitle>
    </CardHeader>
    <CardContent>
      <div className="grid md:grid-cols-2 gap-4">
        <AnimatePresence>
          {agents.map((agent, index) => (
            <motion.div
              key={agent.id}
              initial={{ opacity: 0, scale: 0.9 }}
              animate={{ opacity: 1, scale: 1 }}
              exit={{ opacity: 0, scale: 0.9 }}
              transition={{ delay: index * 0.1 }}
              className="p-4 bg-white/5 rounded-lg border border-white/10
hover:border-orange-500/30 transition-all duration-300"
            >
              <div className="flex items-start justify-between mb-3">
                <div>
```

```
              <h3 className="font-semibold text-white">{agent.name}</h3>
              <Badge
                variant="outline"
                className={`mt-1 ${frameworkColors[agent.framework] ||
frameworkColors.default}`}
              >
                {agent.framework}
              </Badge>
            </div>
            <div className="flex items-center gap-2">
              <div className={`w-3 h-3 rounded-full ${statusColors[agent.status]}
animate-pulse`} />
              <DropdownMenu>
                <DropdownMenuTrigger asChild>
                  <Button variant="ghost" size="icon" className="w-6 h-6">
                    <MoreHorizontal className="w-4 h-4" />
                  </Button>
                </DropdownMenuTrigger>
                <DropdownMenuContent>
                  <DropdownMenuItem>View Details</DropdownMenuItem>
                  <DropdownMenuItem>Configure</DropdownMenuItem>
                  <DropdownMenuItem>Reset</DropdownMenuItem>
                </DropdownMenuContent>
              </DropdownMenu>
            </div>
          </div>

          <div className="space-y-2 mb-4">
            <div className="flex justify-between text-sm">
              <span className="text-gray-400">Performance</span>
              <span className="text-white font-
medium">{agent.performance_score || 0}%</span>
            </div>
            <Progress
              value={agent.performance_score || 0}
              className="h-2 bg-white/10"
            />
          </div>

          <div className="flex gap-2">
            <Button
              size="sm"
              variant="outline"
              className="flex-1 border-green-500/30 text-green-400 hover:bg-
```

```
green-500/10"
                >
                  <Play className="w-3 h-3 mr-1" />
                  Start
                </Button>
                <Button
                  size="sm"
                  variant="outline"
                  className="flex-1 border-yellow-500/30 text-yellow-400 hover:bg-
yellow-500/10"
                >
                  <Pause className="w-3 h-3 mr-1" />
                  Pause
                </Button>
              </div>

              {agent.task_description && (
                <p className="text-xs text-gray-400 mt-2 truncate">
                  Current: {agent.task_description}
                </p>
              )}
            </motion.div>
          ))}
        </AnimatePresence>
      </div>

      {agents.length === 0 && (
        <div className="text-center py-8">
          <Users className="w-12 h-12 text-gray-500 mx-auto mb-4" />
          <p className="text-gray-400">No agents deployed yet</p>
          <Button className="mt-3 bg-orange-500 hover:bg-orange-600">
            Deploy First Agent
          </Button>
        </div>
      )}
    </CardContent>
  </Card>
 );
}

****TaskQueue****

import React from "react";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/
```

```
card";
import { Badge } from "@/components/ui/badge";
import { Clock, CheckCircle, AlertCircle, Loader } from "lucide-react";
import { motion } from "framer-motion";
import { Skeleton } from "@/components/ui/skeleton";

const priorityColors = {
  low: "bg-blue-500/20 text-blue-400 border-blue-500/30",
  medium: "bg-yellow-500/20 text-yellow-400 border-yellow-500/30",
  high: "bg-orange-500/20 text-orange-400 border-orange-500/30",
  critical: "bg-red-500/20 text-red-400 border-red-500/30"
};

const statusIcons = {
  pending: Clock,
  in_progress: Loader,
  completed: CheckCircle,
  failed: AlertCircle
};

export default function TaskQueue({ tasks, isLoading }) {
  if (isLoading) {
    return (
      <Card className="glass-effect border-white/10">
        <CardHeader>
          <CardTitle className="text-white">Task Queue</CardTitle>
        </CardHeader>
        <CardContent>
          <div className="space-y-3">
            {Array(3).fill(0).map((_, i) => (
              <div key={i} className="p-3 bg-white/5 rounded-lg">
                <Skeleton className="h-4 w-3/4 mb-2" />
                <Skeleton className="h-3 w-1/2" />
              </div>
            ))}
          </div>
        </CardContent>
      </Card>
    );
  }

  return (
    <Card className="glass-effect border-white/10">
      <CardHeader>
```

```
      <CardTitle className="flex items-center gap-2 text-white">
        <Clock className="w-5 h-5 text-orange-400" />
        Task Queue
        <Badge variant="outline" className="ml-auto border-orange-500/30 text-
orange-400">
          {tasks.filter(t => t.status === 'pending' || t.status === 'in_progress').length}
Active
        </Badge>
      </CardTitle>
    </CardHeader>
    <CardContent>
      <div className="space-y-3 max-h-80 overflow-y-auto">
        {tasks.slice(0, 8).map((task, index) => {
          const StatusIcon = statusIcons[task.status] || Clock;
          return (
            <motion.div
              key={task.id}
              initial={{ opacity: 0, x: -20 }}
              animate={{ opacity: 1, x: 0 }}
              transition={{ delay: index * 0.05 }}
              className="p-3 bg-white/5 rounded-lg border border-white/10
hover:border-orange-500/20 transition-all duration-300"
            >
              <div className="flex items-start justify-between mb-2">
                <h4 className="font-medium text-white text-sm truncate
flex-1">{task.title}</h4>
                <StatusIcon className={`w-4 h-4 ml-2 ${
                  task.status === 'completed' ? 'text-green-400' :
                  task.status === 'in_progress' ? 'text-orange-400 animate-spin' :
                  task.status === 'failed' ? 'text-red-400' :
                  'text-gray-400'
                }`} />
              </div>
              <div className="flex items-center gap-2">
                <Badge
                  variant="outline"
                  className={`text-xs ${priorityColors[task.priority] ||
priorityColors.medium}`}
                >
                  {task.priority}
                </Badge>
                <Badge variant="outline" className="text-xs border-gray-500/30
text-gray-400">
                  {task.framework}
```

```
              </Badge>
            </div>
            {task.progress > 0 && (
              <div className="mt-2 bg-white/10 rounded-full h-1">
                <div
                  className="bg-orange-400 h-1 rounded-full transition-all
duration-500"
                  style={{ width: `${task.progress}%` }}
                />
              </div>
            )}
          </motion.div>
        );
      })}
    </div>

    {tasks.length === 0 && (
      <div className="text-center py-6">
        <Clock className="w-8 h-8 text-gray-500 mx-auto mb-2" />
        <p className="text-gray-400 text-sm">No tasks in queue</p>
      </div>
    )}
  </CardContent>
 </Card>
 );
}
```

Framework Selector

```
import React, { useState } from "react";
import { Card, CardContent, CardHeader, CardTitle } from "@/components/ui/
card";
import { Button } from "@/components/ui/button";
import { Badge } from "@/components/ui/badge";
import { Settings, ChevronDown } from "lucide-react";
import { motion } from "framer-motion";
import {
  DropdownMenu,
  DropdownMenuContent,
  DropdownMenuItem,
  DropdownMenuTrigger,
} from "@/components/ui/dropdown-menu";

const frameworks = [
```

```
  { id: "autogen", name: "AutoGen", status: "active", color: "bg-blue-500" },
  { id: "crewai", name: "CrewAI", status: "active", color: "bg-purple-500" },
  { id: "langgraph", name: "LangGraph", status: "active", color: "bg-green-500" },
  { id: "cerebras", name: "Cerebras", status: "active", color: "bg-orange-500" },
  { id: "autogpt", name: "AutoGPT", status: "inactive", color: "bg-gray-500" },
  { id: "babyagi", name: "BabyAGI", status: "inactive", color: "bg-gray-500" },
];

export default function FrameworkSelector() {
  const [selectedFramework, setSelectedFramework] = useState("cerebras");

  return (
    <Card className="glass-effect border-white/10">
      <CardHeader>
        <CardTitle className="flex items-center gap-2 text-white">
          <Settings className="w-5 h-5 text-orange-400" />
          Framework Control
        </CardTitle>
      </CardHeader>
      <CardContent>
        <div className="space-y-3">
          <DropdownMenu>
            <DropdownMenuTrigger asChild>
              <Button
                variant="outline"
                className="w-full justify-between border-orange-500/30 text-
orange-400 hover:bg-orange-500/10"
              >
                Primary Framework: {frameworks.find(f => f.id ===
selectedFramework)?.name}
                <ChevronDown className="w-4 h-4" />
              </Button>
            </DropdownMenuTrigger>
            <DropdownMenuContent className="w-full">
              {frameworks.filter(f => f.status === "active").map((framework) => (
                <DropdownMenuItem
                  key={framework.id}
                  onClick={() => setSelectedFramework(framework.id)}
                >
                  <div className={`w-2 h-2 rounded-full ${framework.color} mr-2`} />
                  {framework.name}
                </DropdownMenuItem>
              ))}
```

```
          </DropdownMenuContent>
        </DropdownMenu>

        <div className="space-y-2">
          <p className="text-xs text-gray-400 uppercase tracking-
wider">Available Frameworks</p>
          {frameworks.map((framework, index) => (
            <motion.div
              key={framework.id}
              initial={{ opacity: 0, x: -10 }}
              animate={{ opacity: 1, x: 0 }}
              transition={{ delay: index * 0.05 }}
              className="flex items-center justify-between p-2 bg-white/5 rounded
border border-white/10"
            >
              <div className="flex items-center gap-2">
                <div className={`w-2 h-2 rounded-full ${framework.color}`} />
                <span className="text-sm text-white">{framework.name}</span>
              </div>
              <Badge
                variant="outline"
                className={framework.status === "active"
                  ? "border-green-500/30 text-green-400"
                  : "border-gray-500/30 text-gray-400"
                }
              >
                {framework.status}
              </Badge>
            </motion.div>
          ))}
        </div>

        <Button
          variant="outline"
          className="w-full mt-4 border-orange-500/30 text-orange-400
hover:bg-orange-500/10"
        >
          Configure Frameworks
        </Button>
      </div>
    </CardContent>
  </Card>
 );
```

```
}
```

******AgentCard*****

```
import React from "react";
import { Card, CardContent, CardHeader } from "@/components/ui/card";
import { Badge } from "@/components/ui/badge";
import { Button } from "@/components/ui/button";
import { Progress } from "@/components/ui/progress";
import {
  Brain,
  Play,
  Pause,
  Square,
  Settings,
  MoreHorizontal,
  Clock,
  Zap
} from "lucide-react";
import { motion } from "framer-motion";
import {
  DropdownMenu,
  DropdownMenuContent,
  DropdownMenuItem,
  DropdownMenuTrigger,
} from "@/components/ui/dropdown-menu";

const frameworkColors = {
  autogen: "bg-blue-500/20 text-blue-400 border-blue-500/30",
  crewai: "bg-purple-500/20 text-purple-400 border-purple-500/30",
  langgraph: "bg-green-500/20 text-green-400 border-green-500/30",
  cerebras: "bg-orange-500/20 text-orange-400 border-orange-500/30",
  metagpt: "bg-pink-500/20 text-pink-400 border-pink-500/30",
  autogpt: "bg-cyan-500/20 text-cyan-400 border-cyan-500/30",
  default: "bg-gray-500/20 text-gray-400 border-gray-500/30"
};

const statusColors = {
  running: "bg-green-400",
  idle: "bg-gray-400",
  paused: "bg-yellow-400",
  error: "bg-red-400",
```

```
  completed: "bg-blue-400"
};

export default function AgentCard({ agent, onAction }) {
  const getActionButtons = () => {
    switch (agent.status) {
      case 'running':
        return (
          <>
            <Button
              size="sm"
              variant="outline"
              onClick={() => onAction(agent.id, 'pause')}
              className="border-yellow-500/30 text-yellow-400 hover:bg-
yellow-500/10"
            >
              <Pause className="w-3 h-3 mr-1" />
              Pause
            </Button>
            <Button
              size="sm"
              variant="outline"
              onClick={() => onAction(agent.id, 'stop')}
              className="border-red-500/30 text-red-400 hover:bg-red-500/10"
            >
              <Square className="w-3 h-3 mr-1" />
              Stop
            </Button>
          </>
        );
      case 'paused':
        return (
          <>
            <Button
              size="sm"
              variant="outline"
              onClick={() => onAction(agent.id, 'start')}
              className="border-green-500/30 text-green-400 hover:bg-
green-500/10"
            >
              <Play className="w-3 h-3 mr-1" />
              Resume
            </Button>
            <Button
```

```
            size="sm"
            variant="outline"
            onClick={() => onAction(agent.id, 'stop')}
            className="border-red-500/30 text-red-400 hover:bg-red-500/10"
          >
            <Square className="w-3 h-3 mr-1" />
            Stop
          </Button>
        </>
      );
    default:
      return (
        <Button
          size="sm"
          variant="outline"
          onClick={() => onAction(agent.id, 'start')}
          className="border-green-500/30 text-green-400 hover:bg-green-500/10
flex-1"
        >
          <Play className="w-3 h-3 mr-1" />
          Start
        </Button>
      );
  }
 };

 return (
  <motion.div
   whileHover={{ scale: 1.02 }}
   transition={{ duration: 0.2 }}
  >
   <Card className="glass-effect border-white/10 hover:border-orange-500/30
transition-all duration-300 h-full">
     <CardHeader className="pb-3">
      <div className="flex items-start justify-between">
       <div className="flex items-center gap-3">
        <div className="w-10 h-10 bg-gradient-to-r from-orange-500 to-
orange-600 rounded-lg flex items-center justify-center">
         <Brain className="w-5 h-5 text-white" />
        </div>
        <div>
         <h3 className="font-semibold text-white">{agent.name}</h3>
         <Badge
          variant="outline"
```

```
              className={`mt-1 ${frameworkColors[agent.framework] ||
frameworkColors.default}`}
            >
              {agent.framework}
            </Badge>
          </div>
        </div>
        <div className="flex items-center gap-2">
          <div className={`w-3 h-3 rounded-full ${statusColors[agent.status]} ${
            agent.status === 'running' ? 'animate-pulse' : ''
          }`} />
          <DropdownMenu>
            <DropdownMenuTrigger asChild>
              <Button variant="ghost" size="icon" className="w-6 h-6">
                <MoreHorizontal className="w-4 h-4" />
              </Button>
            </DropdownMenuTrigger>
            <DropdownMenuContent>
              <DropdownMenuItem>View Logs</DropdownMenuItem>
              <DropdownMenuItem>Configure</DropdownMenuItem>
              <DropdownMenuItem>Export Data</DropdownMenuItem>
              <DropdownMenuItem className="text-red-400">Delete Agent</
DropdownMenuItem>
            </DropdownMenuContent>
          </DropdownMenu>
        </div>
      </div>
    </CardHeader>

    <CardContent className="space-y-4">
      {/* Performance Metrics */}
      <div className="space-y-2">
        <div className="flex justify-between text-sm">
          <span className="text-gray-400">Performance</span>
          <span className="text-white font-medium">{agent.performance_score ||
0}%</span>
        </div>
        <Progress
          value={agent.performance_score || 0}
          className="h-2 bg-white/10"
        />
      </div>

      {/* Stats */}
```

```
        <div className="grid grid-cols-2 gap-3">
          <div className="p-2 bg-white/5 rounded border border-white/10">
            <div className="flex items-center gap-2 text-xs text-gray-400">
              <Clock className="w-3 h-3" />
              Execution Time
            </div>
            <div className="text-sm font-medium text-white mt-1">
              {agent.execution_time ? `${agent.execution_time}s` : '--'}
            </div>
          </div>
          <div className="p-2 bg-white/5 rounded border border-white/10">
            <div className="flex items-center gap-2 text-xs text-gray-400">
              <Zap className="w-3 h-3" />
              Memory Usage
            </div>
            <div className="text-sm font-medium text-white mt-1">
              {agent.memory_usage ? `${agent.memory_usage}MB` : '--'}
            </div>
          </div>
        </div>

        {/* Current Task */}
        {agent.task_description && (
          <div className="p-3 bg-white/5 rounded border border-white/10">
            <div className="text-xs text-gray-400 mb-1">Current Task</div>
            <div className="text-sm text-white line-clamp-2">
              {agent.task_description}
            </div>
          </div>
        )}

        {/* Action Buttons */}
        <div className="flex gap-2 pt-2">
          {getActionButtons()}
        </div>
      </CardContent>
    </Card>
  </motion.div>
 );
}
```

CreateAgentModal**

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

```
import {
  Dialog,
  DialogContent,
  DialogHeader,
  DialogTitle,
} from "@/components/ui/dialog";
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/select";
import { Brain, Loader2 } from "lucide-react";

const frameworks = [
  { id: "autogen", name: "AutoGen", description: "Multi-agent conversation
framework" },
  { id: "crewai", name: "CrewAI", description: "Collaborative AI agent framework" },
  { id: "langgraph", name: "LangGraph", description: "Language model workflow
orchestration" },
  { id: "cerebras", name: "Cerebras", description: "High-performance AI
inference" },
  { id: "metagpt", name: "MetaGPT", description: "Multi-agent software
development" },
  { id: "autogpt", name: "AutoGPT", description: "Autonomous AI agent" },
  { id: "babyagi", name: "BabyAGI", description: "Task-driven autonomous agent" },
  { id: "camelai", name: "CamelAI", description: "Communicative agent
framework" },
];

const models = [
  "llama3.1-8b",
  "llama3.1-70b",
  "llama3.1-405b",
  "mixtral-8x7b",
  "gemma-7b",
  "custom"
];
```

```
export default function CreateAgentModal({ isOpen, onClose, onSubmit }) {
  const [formData, setFormData] = useState({
    name: "",
    framework: "",
    model: "",
    task_description: "",
    configuration: {}
  });
  const [isSubmitting, setIsSubmitting] = useState(false);

  const handleSubmit = async (e) => {
    e.preventDefault();
    if (!formData.name || !formData.framework) return;

    setIsSubmitting(true);
    try {
      await onSubmit({
        ...formData,
        status: "idle",
        performance_score: 0,
        last_activity: new Date().toISOString()
      });
      setFormData({
        name: "",
        framework: "",
        model: "",
        task_description: "",
        configuration: {}
      });
    } catch (error) {
      console.error('Error creating agent:', error);
    }
    setIsSubmitting(false);
  };

  const selectedFramework = frameworks.find(f => f.id === formData.framework);

  return (
    <Dialog open={isOpen} onOpenChange={onClose}>
      <DialogContent className="sm:max-w-2xl glass-effect border-white/20">
        <DialogHeader>
          <DialogTitle className="flex items-center gap-2 text-white text-xl">
            <Brain className="w-6 h-6 text-orange-400" />
            Deploy New AI Agent
```

```
          </DialogTitle>
        </DialogHeader>

        <form onSubmit={handleSubmit} className="space-y-6 mt-6">
          {/* Agent Name */}
          <div className="space-y-2">
            <Label className="text-white font-medium">Agent Name</Label>
            <Input
              placeholder="Enter agent name (e.g., Data Analyst Agent)"
              value={formData.name}
              onChange={(e) => setFormData({ ...formData, name: e.target.value })}
              className="bg-white/5 border-white/10 text-white placeholder:text-
gray-400 focus:border-orange-500/50"
              required
            />
          </div>

          {/* Framework Selection */}
          <div className="space-y-2">
            <Label className="text-white font-medium">AI Framework</Label>
            <Select
              value={formData.framework}
              onValueChange={(value) => setFormData({ ...formData, framework:
value })}
            >
              <SelectTrigger className="bg-white/5 border-white/10 text-white">
                <SelectValue placeholder="Select an AI framework" />
              </SelectTrigger>
              <SelectContent>
                {frameworks.map((framework) => (
                  <SelectItem key={framework.id} value={framework.id}>
                    <div>
                      <div className="font-medium">{framework.name}</div>
                      <div className="text-xs text-gray-400">{framework.description}</
div>
                    </div>
                  </SelectItem>
                ))}
              </SelectContent>
            </Select>
            {selectedFramework && (
              <p className="text-sm text-gray-400">{selectedFramework.description}
</p>
```

```
          )}
        </div>

        {/* Model Selection */}
        <div className="space-y-2">
          <Label className="text-white font-medium">AI Model</Label>
          <Select
            value={formData.model}
            onValueChange={(value) => setFormData({ ...formData, model: value })}
          >
            <SelectTrigger className="bg-white/5 border-white/10 text-white">
              <SelectValue placeholder="Select AI model" />
            </SelectTrigger>
            <SelectContent>
              {models.map((model) => (
                <SelectItem key={model} value={model}>
                  {model}
                </SelectItem>
              ))}
            </SelectContent>
          </Select>
        </div>

        {/* Task Description */}
        <div className="space-y-2">
          <Label className="text-white font-medium">Initial Task (Optional)</
Label>
          <Textarea
            placeholder="Describe the initial task for this agent..."
            value={formData.task_description}
            onChange={(e) => setFormData({ ...formData, task_description:
e.target.value })}
            className="bg-white/5 border-white/10 text-white placeholder:text-
gray-400 focus:border-orange-500/50 h-24"
          />
        </div>

        {/* Action Buttons */}
        <div className="flex justify-end gap-3 pt-4">
          <Button
            type="button"
            variant="outline"
            onClick={onClose}
            className="border-white/20 text-gray-300 hover:bg-white/5"
```

```
            >
              Cancel
            </Button>
            <Button
              type="submit"
              disabled={!formData.name || !formData.framework || isSubmitting}
              className="bg-orange-500 hover:bg-orange-600 text-white"
            >
              {isSubmitting ? (
                <>
                  <Loader2 className="w-4 h-4 mr-2 animate-spin" />
                  Deploying...
                </>
              ) : (
                <>
                  <Brain className="w-4 h-4 mr-2" />
                  Deploy Agent
                </>
              )}
            </Button>
          </div>
        </form>
      </DialogContent>
    </Dialog>
  );
}
```

AgentFilters

```
import React from "react";
import {
  Select,
  SelectContent,
  SelectItem,
  SelectTrigger,
  SelectValue,
} from "@/components/ui/select";
import { Filter } from "lucide-react";

const frameworks = [
  { id: "all", name: "All Frameworks" },
```

```
  { id: "autogen", name: "AutoGen" },
  { id: "crewai", name: "CrewAI" },
  { id: "langgraph", name: "LangGraph" },
  { id: "cerebras", name: "Cerebras" },
  { id: "metagpt", name: "MetaGPT" },
  { id: "autogpt", name: "AutoGPT" },
];

const statuses = [
  { id: "all", name: "All Status" },
  { id: "running", name: "Running" },
  { id: "idle", name: "Idle" },
  { id: "paused", name: "Paused" },
  { id: "error", name: "Error" },
];

export default function AgentFilters({
  selectedFramework,
  selectedStatus,
  onFrameworkChange,
  onStatusChange
}) {
  return (
    <div className="flex gap-3">
      <div className="flex items-center gap-2">
        <Filter className="w-4 h-4 text-gray-400" />
        <Select value={selectedFramework} onValueChange={onFrameworkChange}>
          <SelectTrigger className="w-40 bg-white/5 border-white/10 text-white">
            <SelectValue />
          </SelectTrigger>
          <SelectContent>
            {frameworks.map((framework) => (
              <SelectItem key={framework.id} value={framework.id}>
                {framework.name}
              </SelectItem>
            ))}
          </SelectContent>
        </Select>
      </div>

      <Select value={selectedStatus} onValueChange={onStatusChange}>
        <SelectTrigger className="w-32 bg-white/5 border-white/10 text-white">
          <SelectValue />
        </SelectTrigger>
```

```
      <SelectContent>
        {statuses.map((status) => (
          <SelectItem key={status.id} value={status.id}>
            {status.name}
          </SelectItem>
        ))}
      </SelectContent>
    </Select>
  </div>
 );
}
```

Agent

```
{
  "name": "Agent",
  "type": "object",
  "properties": {
    "name": {
      "type": "string",
      "description": "Agent name/identifier"
    },
    "framework": {
      "type": "string",
      "enum": [
        "autogen",
        "metagpt",
        "crewai",
        "autogpt",
        "babyagi",
        "langgraph",
        "camelai",
        "agentverse",
        "openagents",
        "miniagi",
        "orca",
        "cerebras",
        "cerebras_autogen"
      ],
      "description": "AI framework being used"
    },
    "status": {
      "type": "string",
      "enum": [
```

```
        "idle",
        "running",
        "paused",
        "completed",
        "error"
      ],
      "default": "idle",
      "description": "Current agent status"
    },
    "model": {
      "type": "string",
      "description": "AI model being used (e.g., Cerebras model)"
    },
    "task_description": {
      "type": "string",
      "description": "Current or last task description"
    },
    "performance_score": {
      "type": "number",
      "minimum": 0,
      "maximum": 100,
      "description": "Performance rating 0-100"
    },
    "execution_time": {
      "type": "number",
      "description": "Last execution time in seconds"
    },
    "memory_usage": {
      "type": "number",
      "description": "Memory usage in MB"
    },
    "configuration": {
      "type": "object",
      "description": "Framework-specific configuration"
    },
    "last_activity": {
      "type": "string",
      "format": "date-time",
      "description": "Last activity timestamp"
    }
  },
  "required": [
    "name",
    "framework"
```

```
 ]
}
```

Agent*

```
{
  "name": "Agent",
  "type": "object",
  "properties": {
    "name": {
      "type": "string",
      "description": "Agent name/identifier"
    },
    "framework": {
      "type": "string",
      "enum": [
        "autogen",
        "metagpt",
        "crewai",
        "autogpt",
        "babyagi",
        "langgraph",
        "camelai",
        "agentverse",
        "openagents",
        "miniagi",
        "orca",
        "cerebras",
        "cerebras_autogen"
      ],
      "description": "AI framework being used"
    },
    "status": {
      "type": "string",
      "enum": [
        "idle",
        "running",
        "paused",
        "completed",
        "error"
      ],
      "default": "idle",
```

```
      "description": "Current agent status"
    },
    "model": {
      "type": "string",
      "description": "AI model being used (e.g., Cerebras model)"
    },
    "task_description": {
      "type": "string",
      "description": "Current or last task description"
    },
    "performance_score": {
      "type": "number",
      "minimum": 0,
      "maximum": 100,
      "description": "Performance rating 0-100"
    },
    "execution_time": {
      "type": "number",
      "description": "Last execution time in seconds"
    },
    "memory_usage": {
      "type": "number",
      "description": "Memory usage in MB"
    },
    "configuration": {
      "type": "object",
      "description": "Framework-specific configuration"
    },
    "last_activity": {
      "type": "string",
      "format": "date-time",
      "description": "Last activity timestamp"
    }
  },
  "required": [
    "name",
    "framework"
  ]
}

*****Task******

{
  "name": "Task",
```

```
"type": "object",
"properties": {
 "title": {
  "type": "string",
  "description": "Task title"
 },
 "description": {
  "type": "string",
  "description": "Detailed task description"
 },
 "agent_id": {
  "type": "string",
  "description": "ID of assigned agent"
 },
 "framework": {
  "type": "string",
  "enum": [
   "autogen",
   "metagpt",
   "crewai",
   "autogpt",
   "babyagi",
   "langgraph",
   "camelai",
   "agentverse",
   "openagents",
   "miniagi",
   "orca",
   "cerebras",
   "cerebras_autogen"
  ],
  "description": "Framework used for this task"
 },
 "status": {
  "type": "string",
  "enum": [
   "pending",
   "in_progress",
   "completed",
   "failed",
   "cancelled"
  ],
  "default": "pending",
  "description": "Task status"
```

```
  },
  "priority": {
    "type": "string",
    "enum": [
      "low",
      "medium",
      "high",
      "critical"
    ],
    "default": "medium",
    "description": "Task priority"
  },
  "progress": {
    "type": "number",
    "minimum": 0,
    "maximum": 100,
    "default": 0,
    "description": "Task completion percentage"
  },
  "start_time": {
    "type": "string",
    "format": "date-time",
    "description": "Task start time"
  },
  "completion_time": {
    "type": "string",
    "format": "date-time",
    "description": "Task completion time"
  },
  "result": {
    "type": "string",
    "description": "Task execution result or output"
  },
  "error_message": {
    "type": "string",
    "description": "Error message if task failed"
  },
  "estimated_duration": {
    "type": "number",
    "description": "Estimated duration in minutes"
  }
},
"required": [
  "title",
```

```
    "description"
  ]
}
```

API key*

```
{
  "name": "APIKey",
  "type": "object",
  "properties": {
    "name": {
      "type": "string",
      "description": "Friendly name for the API key"
    },
    "provider": {
      "type": "string",
      "enum": [
        "cerebras",
        "openai",
        "anthropic",
        "google",
        "cohere",
        "custom"
      ],
      "description": "API provider"
    },
    "key_value": {
      "type": "string",
      "description": "Encrypted API key value"
    },
    "status": {
      "type": "string",
      "enum": [
        "active",
        "inactive",
        "expired",
        "invalid"
      ],
      "default": "active",
      "description": "API key status"
    },
    "last_used": {
      "type": "string",
      "format": "date-time",
```

```
      "description": "Last time this key was used"
    },
    "usage_count": {
      "type": "number",
      "default": 0,
      "description": "Number of times this key has been used"
    },
    "rate_limit": {
      "type": "number",
      "description": "Requests per minute limit"
    },
    "expires_at": {
      "type": "string",
      "format": "date-time",
      "description": "Key expiration date"
    }
  },
  "required": [
    "name",
    "provider",
    "key_value"
  ]
}
```

Layout.js

```
import React from "react";
import { Link, useLocation } from "react-router-dom";
import { createPageUrl } from "@/utils";
import {
  Brain,
  LayoutDashboard,
  Users,
  Terminal,
  BarChart3,
  Settings,
  Zap,
  Activity
} from "lucide-react";
import {
  Sidebar,
  SidebarContent,
  SidebarGroup,
  SidebarGroupContent,
```

```
  SidebarGroupLabel,
  SidebarMenu,
  SidebarMenuButton,
  SidebarMenuItem,
  SidebarHeader,
  SidebarFooter,
  SidebarProvider,
  SidebarTrigger,
} from "@/components/ui/sidebar";

const navigationItems = [
  {
    title: "Orchestration Hub",
    url: createPageUrl("Dashboard"),
    icon: LayoutDashboard,
  },
  {
    title: "Agent Management",
    url: createPageUrl("Agents"),
    icon: Users,
  },
  {
    title: "Task Console",
    url: createPageUrl("Console"),
    icon: Terminal,
  },
  {
    title: "Analytics",
    url: createPageUrl("Analytics"),
    icon: BarChart3,
  },
  {
    title: "API Configuration",
    url: createPageUrl("Settings"),
    icon: Settings,
  },
];

export default function Layout({ children, currentPageName }) {
  const location = useLocation();

  return (
    <SidebarProvider>
      <style>{`
```

```
      :root {
        --primary-bg: #0a0a0a;
        --secondary-bg: #1a1a1a;
        --accent-orange: #ff6b35;
        --accent-orange-hover: #ff8c42;
        --text-primary: #ffffff;
        --text-secondary: #a1a1aa;
        --border-color: #2a2a2a;
      }

      body {
        background: var(--primary-bg);
        color: var(--text-primary);
      }

      .glass-effect {
        background: rgba(26, 26, 26, 0.8);
        backdrop-filter: blur(10px);
        border: 1px solid rgba(255, 255, 255, 0.1);
      }

      .glow-orange {
        box-shadow: 0 0 20px rgba(255, 107, 53, 0.3);
      }

      .gradient-text {
        background: linear-gradient(135deg, #ff6b35, #ff8c42);
        -webkit-background-clip: text;
        -webkit-text-fill-color: transparent;
        background-clip: text;
      }
    `}</style>

    <div className="min-h-screen flex w-full" style={{background: 'var(--primary-bg)'}}>
      <Sidebar className="border-r" style={{borderColor: 'var(--border-color)', background: 'var(--secondary-bg)'}}>
        <SidebarHeader className="border-b p-6" style={{borderColor: 'var(--border-color)'}}>
          <div className="flex items-center gap-3">
            <div className="w-10 h-10 bg-gradient-to-r from-orange-500 to-orange-600 rounded-xl flex items-center justify-center glow-orange">
              <Brain className="w-6 h-6 text-white" />
            </div>
```

```
            <div>
              <h2 className="font-bold text-xl gradient-text">CerebralOps</h2>
              <p className="text-xs opacity-70">AI Agent Orchestration</p>
            </div>
          </div>
        </SidebarHeader>

        <SidebarContent className="p-4">
          <SidebarGroup>
            <SidebarGroupLabel className="text-xs font-medium uppercase
tracking-wider px-2 py-3 opacity-70">
              Navigation
            </SidebarGroupLabel>
            <SidebarGroupContent>
              <SidebarMenu>
                {navigationItems.map((item) => (
                  <SidebarMenuItem key={item.title}>
                    <SidebarMenuButton
                      asChild
                      className={`transition-all duration-300 rounded-xl mb-1 border-0
${
                        location.pathname === item.url
                          ? 'bg-orange-500/20 text-orange-400 shadow-lg'
                          : 'hover:bg-white/5 text-gray-300 hover:text-orange-400'
                      }`}
                    >
                      <Link to={item.url} className="flex items-center gap-3 px-4
py-3">
                        <item.icon className="w-5 h-5" />
                        <span className="font-medium">{item.title}</span>
                      </Link>
                    </SidebarMenuButton>
                  </SidebarMenuItem>
                ))}
              </SidebarMenu>
            </SidebarGroupContent>
          </SidebarGroup>

          <SidebarGroup className="mt-8">
            <SidebarGroupLabel className="text-xs font-medium uppercase
tracking-wider px-2 py-3 opacity-70">
              System Status
            </SidebarGroupLabel>
            <SidebarGroupContent>
```

```
            <div className="px-3 py-2 space-y-3">
              <div className="flex items-center justify-between text-sm">
                <div className="flex items-center gap-2">
                  <div className="w-2 h-2 bg-green-500 rounded-full animate-
pulse"></div>
                  <span className="opacity-70">Cerebras API</span>
                </div>
                <span className="text-green-400 font-medium">Online</span>
              </div>
              <div className="flex items-center justify-between text-sm">
                <div className="flex items-center gap-2">
                  <Activity className="w-4 h-4 opacity-50" />
                  <span className="opacity-70">Active Agents</span>
                </div>
                <span className="font-bold text-orange-400">12</span>
              </div>
              <div className="flex items-center justify-between text-sm">
                <div className="flex items-center gap-2">
                  <Zap className="w-4 h-4 opacity-50" />
                  <span className="opacity-70">Tasks Today</span>
                </div>
                <span className="font-bold text-blue-400">47</span>
              </div>
            </div>
          </SidebarGroupContent>
        </SidebarGroup>
      </SidebarContent>

      <SidebarFooter className="border-t p-4" style={{borderColor: 'var(--
border-color)'}}>
        <div className="flex items-center gap-3">
          <div className="w-8 h-8 bg-gradient-to-r from-purple-500 to-
orange-500 rounded-full flex items-center justify-center">
            <span className="text-white font-medium text-sm">AI</span>
          </div>
          <div className="flex-1 min-w-0">
            <p className="font-medium text-sm truncate">AI Orchestrator</p>
            <p className="text-xs opacity-70 truncate">Neural Operations Active</
p>
          </div>
        </div>
      </SidebarFooter>
    </Sidebar>
```

```
      <main className="flex-1 flex flex-col">
        <header className="bg-black/20 border-b px-6 py-4 md:hidden"
style={{borderColor: 'var(--border-color)'}}>
          <div className="flex items-center gap-4">
            <SidebarTrigger className="hover:bg-white/10 p-2 rounded-lg
transition-colors duration-200" />
            <h1 className="text-xl font-semibold gradient-text">CerebralOps</h1>
          </div>
        </header>

        <div className="flex-1 overflow-auto" style={{background: 'var(--primary-
bg)'}}>
          {children}
        </div>
      </main>
    </div>
  </SidebarProvider>
 );
}
```