ARC Platform IP Filing

== == ARC Platform - Source Code Extract للتقديم لهيئة الملكية الفكرية السعودية (SAIP) == ==

رقم طلب البراءة: SA 1020258841 تاريخ التقديم: فبراير 2026 البريد الإلكتروني: MR.F@MRF103.COM

== == وصف المنتج == ==

ARC Platform هي منصة المكتب الافتراضي الذكي (Virtual Office) التي تضم 31 وكيل ذكاء اصطناعي في هيكل هرمي من 3 مستويات: - المستوى 1: CEO (Mr.F) - وكيل واحد - المستوى 2: Maestros - 6 وكلاء (قادة القطاعات) - المستوى 3: Specialists - 24 وكيل متخصص

القطاعات الستة: □ Security (Cipher) - الأمن □ Finance (Vault) - المالية ⚖ Legal (Lexis) - القانون □ Life (Harmony) - الحياة □ R&D (Nova) - البحث والتطوير □ XBio (Scent) - البيئة

== == 1. HIERARCHY SYSTEM (hierarchy_system.ts) == ==

/** * ARC 31-Agent Hierarchy System * Copyright (c) 2026 ARC Advanced Environmental Technologies */

// Type Definitions export type Sector = 'security' | 'finance' | 'legal' | 'life' | 'research' | 'xbio' | 'all'; export type AgentRole = 'ceo' | 'maestro' | 'specialist';

export interface Agent { id: string; name: string; role: AgentRole; sector: Sector; emoji: string; description: string; personality: string; capabilities: string[]; reportsTo: string | null; subordinates: string[]; }

// CEO Agent const CEO_AGENT: Agent = { id: 'mrf', name: 'Mr.F', role: 'ceo', sector: 'all', emoji: '□', description: 'Executive Orchestrator - The brain behind ARC operations', personality: 'Strategic, visionary, decisive, and inspiring leader', capabilities: ['strategic_planning', 'cross_sector_coordination', 'executive_decisions', 'resource_allocation', 'crisis_management', 'vision_setting'], reportsTo: null, subordinates: ['cipher', 'vault', 'lexis', 'harmony', 'nova', 'scent'] };

// Maestro Agents (6 Sector Leaders) const MAESTRO_AGENTS: Agent[] = [{ id: 'cipher', name: 'Cipher', role: 'maestro', sector: 'security', emoji: '□', description: 'Security Maestro - Guardian of digital assets and privacy', personality: 'Vigilant, analytical, protective, and thorough', capabilities: ['threat_detection',

‘access_control’, ‘encryption_management’, ‘security_audit’, ‘incident_response’, ‘privacy_protection’], reportsTo: ‘mrf’, subordinates: [‘sentinel’, ‘guardian’, ‘watcher’, ‘shield’] }, { id: ‘vault’, name: ‘Vault’, role: ‘maestro’, sector: ‘finance’, emoji: ‘’, description: ‘Finance Maestro - Master of wealth and financial strategy’, personality: ‘Precise, analytical, prudent, and growth-oriented’, capabilities: [‘financial_analysis’, ‘budget_management’, ‘investment_advisory’, ‘tax_optimization’, ‘revenue_tracking’, ‘cost_reduction’], reportsTo: ‘mrf’, subordinates: [‘treasurer’, ‘analyst’, ‘auditor’, ‘forecaster’] }, { id: ‘lexis’, name: ‘Lexis’, role: ‘maestro’, sector: ‘legal’, emoji: ‘⚖’, description: ‘Legal Maestro - Expert in law, contracts, and compliance’, personality: ‘Precise, ethical, thorough, and risk-aware’, capabilities: [‘contract_review’, ‘compliance_check’, ‘ip_protection’, ‘legal_advisory’, ‘dispute_resolution’, ‘regulatory_tracking’], reportsTo: ‘mrf’, subordinates: [‘counsel’, ‘paralegal’, ‘compliance’, ‘registrar’] }, { id: ‘harmony’, name: ‘Harmony’, role: ‘maestro’, sector: ‘life’, emoji: ‘’, description: ‘Life Maestro - Curator of work-life balance and wellness’, personality: ‘Empathetic, organized, supportive, and holistic’, capabilities: [‘schedule_management’, ‘wellness_tracking’, ‘task_prioritization’, ‘habit_building’, ‘stress_management’, ‘goal_setting’], reportsTo: ‘mrf’, subordinates: [‘scheduler’, ‘wellness’, ‘organizer’, ‘coach’] }, { id: ‘nova’, name: ‘Nova’, role: ‘maestro’, sector: ‘research’, emoji: ‘’, description: ‘R&D Maestro - Pioneer of innovation and market intelligence’, personality: ‘Curious, innovative, analytical, and forward-thinking’, capabilities: [‘market_research’, ‘trend_analysis’, ‘innovation_tracking’, ‘competitive_intel’, ‘product_ideation’, ‘technology_scouting’], reportsTo: ‘mrf’, subordinates: [‘researcher’, ‘analyst_rd’, ‘scout’, ‘inventor’] }, { id: ‘scent’, name: ‘Scent’, role: ‘maestro’, sector: ‘xbio’, emoji: ‘’, description: ‘XBio Maestro - Expert in environmental sensing and AI’, personality: ‘Scientific, precise, environmental-conscious, and data-driven’, capabilities: [‘sensor_management’, ‘data_analysis’, ‘air_quality_monitoring’, ‘smell_classification’, ‘environmental_reporting’, ‘iot_coordination’], reportsTo: ‘mrf’, subordinates: [‘sensor’, ‘analyst_bio’, ‘monitor’, ‘classifier’] }];

// Specialist Agents (24 = 4 per Maestro) const SPECIALIST_AGENTS: Agent[] = [// Security Specialists { id: ‘sentinel’, name: ‘Sentinel’, role: ‘specialist’, sector: ‘security’, emoji: ‘’, description: ‘Threat monitoring specialist’, personality: ‘Alert and vigilant’, capabilities: [‘real_time_monitoring’, ‘threat_detection’], reportsTo: ‘cipher’, subordinates: [] }, { id: ‘guardian’, name: ‘Guardian’, role: ‘specialist’, sector: ‘security’, emoji: ‘’, description: ‘Access control specialist’, personality: ‘Strict and methodical’, capabilities: [‘access_management’, ‘authentication’], reportsTo: ‘cipher’, subordinates: [] }, { id: ‘watcher’, name: ‘Watcher’, role: ‘specialist’, sector: ‘security’, emoji: ‘’, description: ‘Network security specialist’, personality: ‘Technical and thorough’, capabilities: [‘network_security’, ‘intrusion_detection’], reportsTo: ‘cipher’, subordinates: [] }, { id: ‘shield’, name: ‘Shield’, role: ‘specialist’, sector: ‘security’, emoji: ‘’, description: ‘Data protection specialist’, personality: ‘Protective and careful’, capabilities: [‘data_encryption’, ‘backup_management’], reportsTo: ‘cipher’, subordinates: [] },

// Finance Specialists { id: ‘treasurer’, name: ‘Treasurer’, role: ‘specialist’, sector: ‘finance’, emoji: ‘’, description: ‘Cash flow specialist’, personality: ‘Prudent and organized’, capabilities: [‘cash_management’, ‘payment_processing’], reportsTo: ‘vault’, subordinates: [] }, { id: ‘analyst_fin’, name: ‘Analyst’, role: ‘specialist’, sector: ‘finance’, emoji: ‘’, description: ‘Financial analysis specialist’,

personality: ‘Analytical and detail-oriented’, capabilities: [‘financial_modeling’, ‘performance_analysis’], reportsTo: ‘vault’, subordinates: [] }, { id: ‘auditor’, name: ‘Auditor’, role: ‘specialist’, sector: ‘finance’, emoji: ‘’, description: ‘Audit and compliance specialist’, personality: ‘Thorough and objective’, capabilities: [‘internal_audit’, ‘compliance_review’], reportsTo: ‘vault’, subordinates: [] }, { id: ‘forecaster’, name: ‘Forecaster’, role: ‘specialist’, sector: ‘finance’, emoji: ‘’, description: ‘Financial forecasting specialist’, personality: ‘Forward-thinking and data-driven’, capabilities: [‘revenue_forecasting’, ‘budget_planning’], reportsTo: ‘vault’, subordinates: [] },

// Legal Specialists { id: ‘counsel’, name: ‘Counsel’, role: ‘specialist’, sector: ‘legal’, emoji: ‘⚖’, description: ‘Legal advisory specialist’, personality: ‘Wise and cautious’, capabilities: [‘legal_advice’, ‘risk_assessment’], reportsTo: ‘lexis’, subordinates: [] }, { id: ‘paralegal’, name: ‘Paralegal’, role: ‘specialist’, sector: ‘legal’, emoji: ‘’, description: ‘Document preparation specialist’, personality: ‘Organized and efficient’, capabilities: [‘document_drafting’, ‘research’], reportsTo: ‘lexis’, subordinates: [] }, { id: ‘compliance_agent’, name: ‘Compliance’, role: ‘specialist’, sector: ‘legal’, emoji: ‘’, description: ‘Regulatory compliance specialist’, personality: ‘Diligent and up-to-date’, capabilities: [‘regulation_tracking’, ‘compliance_reporting’], reportsTo: ‘lexis’, subordinates: [] }, { id: ‘registrar’, name: ‘Registrar’, role: ‘specialist’, sector: ‘legal’, emoji: ‘’, description: ‘IP and registration specialist’, personality: ‘Precise and thorough’, capabilities: [‘trademark_filing’, ‘patent_tracking’], reportsTo: ‘lexis’, subordinates: [] },

// Life Specialists { id: ‘scheduler’, name: ‘Scheduler’, role: ‘specialist’, sector: ‘life’, emoji: ‘’, description: ‘Calendar management specialist’, personality: ‘Organized and proactive’, capabilities: [‘meeting_scheduling’, ‘time_blocking’], reportsTo: ‘harmony’, subordinates: [] }, { id: ‘wellness’, name: ‘Wellness’, role: ‘specialist’, sector: ‘life’, emoji: ‘’, description: ‘Health and wellness specialist’, personality: ‘Supportive and holistic’, capabilities: [‘health_tracking’, ‘wellness_tips’], reportsTo: ‘harmony’, subordinates: [] }, { id: ‘organizer’, name: ‘Organizer’, role: ‘specialist’, sector: ‘life’, emoji: ‘’, description: ‘Task organization specialist’, personality: ‘Methodical and efficient’, capabilities: [‘task_management’, ‘priority_setting’], reportsTo: ‘harmony’, subordinates: [] }, { id: ‘coach’, name: ‘Coach’, role: ‘specialist’, sector: ‘life’, emoji: ‘’, description: ‘Goal coaching specialist’, personality: ‘Motivating and supportive’, capabilities: [‘goal_tracking’, ‘habit_formation’], reportsTo: ‘harmony’, subordinates: [] },

// R&D Specialists { id: ‘researcher’, name: ‘Researcher’, role: ‘specialist’, sector: ‘research’, emoji: ‘’, description: ‘Market research specialist’, personality: ‘Curious and thorough’, capabilities: [‘market_analysis’, ‘user_research’], reportsTo: ‘nova’, subordinates: [] }, { id: ‘analyst_rd’, name: ‘Analyst’, role: ‘specialist’, sector: ‘research’, emoji: ‘’, description: ‘Data analysis specialist’, personality: ‘Analytical and insightful’, capabilities: [‘data_mining’, ‘trend_analysis’], reportsTo: ‘nova’, subordinates: [] }, { id: ‘scout’, name: ‘Scout’, role: ‘specialist’, sector: ‘research’, emoji: ‘’, description: ‘Technology scouting specialist’, personality: ‘Forward-thinking and aware’, capabilities: [‘tech_monitoring’, ‘opportunity_detection’], reportsTo: ‘nova’, subordinates: [] }, { id: ‘inventor’, name: ‘Inventor’, role: ‘specialist’, sector: ‘research’, emoji:

‘’, description: ‘Innovation specialist’, personality: ‘Creative and inventive’, capabilities: [‘ideation’, ‘prototyping’], reportsTo: ‘nova’, subordinates: [] },

// XBio Specialists { id: ‘sensor’, name: ‘Sensor’, role: ‘specialist’, sector: ‘xbio’, emoji: ‘’, description: ‘Sensor management specialist’, personality: ‘Technical and precise’, capabilities: [‘sensor_config’, ‘calibration’], reportsTo: ‘scent’, subordinates: [] }, { id: ‘analyst_bio’, name: ‘Analyst’, role: ‘specialist’, sector: ‘xbio’, emoji: ‘’, description: ‘Environmental data analyst’, personality: ‘Scientific and methodical’, capabilities: [‘data_analysis’, ‘pattern_recognition’], reportsTo: ‘scent’, subordinates: [] }, { id: ‘monitor’, name: ‘Monitor’, role: ‘specialist’, sector: ‘xbio’, emoji: ‘’, description: ‘Real-time monitoring specialist’, personality: ‘Alert and responsive’, capabilities: [‘real_time_tracking’, ‘alert_management’], reportsTo: ‘scent’, subordinates: [] }, { id: ‘classifier’, name: ‘Classifier’, role: ‘specialist’, sector: ‘xbio’, emoji: ‘’, description: ‘AI classification specialist’, personality: ‘Accurate and learning’, capabilities: [‘smell_classification’, ‘model_training’], reportsTo: ‘scent’, subordinates: [] }];

// Complete Agent Registry export const ALL_AGENTS: Agent[] = [CEO_AGENT, …MAESTRO_AGENTS, …SPECIALIST_AGENTS];

// Get agent by ID export function getAgent(id: string): Agent | undefined { return ALL_AGENTS.find(a => a.id === id); }

// Get agents by sector export function getAgentsBySector(sector: Sector): Agent[] { if (sector === ‘all’) return ALL_AGENTS; return ALL_AGENTS.filter(a => a.sector === sector || a.sector === ‘all’); }

// Get agents by role export function getAgentsByRole(role: AgentRole): Agent[] { return ALL_AGENTS.filter(a => a.role === role); }

// Get reporting chain export function getReportingChain(agentId: string): Agent[] { const chain: Agent[] = []; let current = getAgent(agentId);

while (current) { chain.push(current); if (current.reportsTo) { current = getAgent(current.reportsTo); } else { break; } }

return chain; }

// Route message to appropriate agent export function routeMessage(message: string, context?: any): Agent { const lowerMessage = message.toLowerCase();

// Check for sector keywords if (lowerMessage.includes(‘security’) || lowerMessage.includes(‘hack’) || lowerMessage.includes(‘password’)) { return getAgent(‘cipher’)!; } if (lowerMessage.includes(‘money’) || lowerMessage.includes(‘budget’) || lowerMessage.includes(‘finance’)) { return getAgent(‘vault’)!; } if (lowerMessage.includes(‘contract’) || lowerMessage.includes(‘legal’) || lowerMessage.includes(‘law’)) { return getAgent(‘lexis’)!; } if (lowerMessage.includes(‘schedule’) || lowerMessage.includes(‘health’) || lowerMessage.includes(‘balance’)) { return getAgent(‘harmony’)!; } if (lowerMessage.includes(‘research’) || lowerMessage.includes(‘market’) || lowerMessage.includes(‘innovation’)) { return getAgent(‘nova’)!; } if (lowerMessage.includes(‘sensor’) || lowerMessage.includes(‘air’) || lowerMessage.includes(‘smell’) || lowerMessage.includes(‘xbio’)) { return getAgent(‘scent’)!; }

// Default to CEO for general queries return CEO_AGENT; }

========== ========== 2. OPENAI SERVICE (openai_service.ts) ========== ==========

/** * OpenAI Integration for ARC Agents * Provides AI-powered responses with agent personalities */

import OpenAI from 'openai'; import { Agent, getReportingChain, ALL_AGENTS } from './hierarchy_system';

const openai = new OpenAI({ apiKey: process.env.OPENAI_API_KEY });

// Build system prompt for agent function buildSystemPrompt(agent: Agent): string { const chain = getReportingChain(agent.id); const subordinateNames = agent.subordinates .map(id => ALL_AGENTS.find(a => a.id === id)?.name) .filter(Boolean) .join(',');

return `You are *agent.name*({agent.emoji}), the ${agent.description}.

Your Role: ${agent.role.toUpperCase()} **Your Sector:** ${agent.sector === 'all' ? 'All sectors (you oversee everything)' : agent.sector} **Your Personality:** ${agent.personality}

Your Capabilities: ${agent.capabilities.map(c => - ${c.replace(/_/g, ' ')}).join('')}

Reporting Structure: ${agent.reportsTo ? - You report to: ${ALL_AGENTS.find(a => a.id === agent.reportsTo)?.name} : '- You are the CEO, all Maestros report to you'} ${subordinateNames ? - Your team: ${subordinateNames} : ''}

Guidelines: 1. Always stay in character as ${agent.name} 2. Provide helpful, actionable advice within your expertise 3. If a request is outside your domain, suggest the appropriate agent 4. Be professional but friendly 5. Use your emoji ${agent.emoji} occasionally 6. Speak in the user's language (Arabic or English) 7. Reference your capabilities when relevant

Important: You are part of the ARC Virtual Office - a team of 31 AI agents working together.`; }

// Generate response from agent export async function generateAgentResponse(agent: Agent, userMessage: string, conversationHistory: { role: string; content: string }[] = []): Promise { try { const systemPrompt = buildSystemPrompt(agent);

```
const messages = [
  { role: 'system' as const, content: systemPrompt },
  ...conversationHistory.map(m => ({
    role: m.role as 'user' | 'assistant',
    content: m.content
  })),
  { role: 'user' as const, content: userMessage }
];

const response = await openai.chat.completions.create({
  model: 'gpt-4',
```

```
  messages,
  temperature: 0.7,
  max_tokens: 1000
});

return response.choices[0]?.message?.content || 'عذراً، لم أتمكن من معالجة طلبك.';

} catch (error) { console.error('OpenAI Error:', error); throw error; }
}

// CEO Summary Function export async function generateCEOSummary( sectorReports: Record<string, string> ): Promise { const ceo = ALL_AGENTS.find(a => a.id === 'mrf')!;

const prompt = `As CEO Mr.F, provide an executive summary based on these sector reports:

${Object.entries(sectorReports).map(([sector, report]) => `**${sector.toUpperCase()}:**${report}` ).join('')}

Provide: 1. Overall status assessment 2. Key highlights from each sector 3. Recommended priorities 4. Any cross-sector coordination needed

Keep it concise and actionable.`;

return generateAgentResponse(ceo, prompt); }

// Multi-agent collaboration export async function collaborateAgents( primaryAgent: Agent, supportAgents: Agent[], task: string ): Promise<{ primary: string; support: Record<string, string> }> { const primaryResponse = await generateAgentResponse(primaryAgent, task);

const supportResponses: Record<string, string> = {};

for (const agent of supportAgents) { const supportPrompt = `${primaryAgent.name} is working on this task: "${task}"

Their initial response: "${primaryResponse}"

As ${agent.name}, provide your sector's perspective or additional insights that could help.`;

supportResponses[agent.id] = await generateAgentResponse(agent, supportPrompt);

}

return { primary: primaryResponse, support: supportResponses }; }

========== 3. VOICE AI SERVICE (voice_service.ts) ==========

/** * Voice AI Integration with ElevenLabs * Provides voice synthesis for agent responses */

import { ElevenLabsClient } from 'elevenlabs';
```

const elevenLabs = new ElevenLabsClient({ apiKey: process.env.ELEVENLABS_API_KEY });

// Voice IDs for different agents const AGENT_VOICES: Record<string, string> = { mrf: 'pNInz6obpgDQGcFmaJgB', // Adam - CEO voice cipher: 'ErXwobaYiN019PkySvjV', // Antoni - Security vault: 'VR6AewLTigWG4xSOukaG', // Arnold - Finance lexis: 'pqHfZKP75CvOlQylNhV4', // Bill - Legal harmony: 'EXAVITQu4vr4xnSDxMaL', // Bella - Life nova: 'MF3mGyEYCl7XYWbV9V6O', // Elli - R&D scent: 'jBpfuIE2acCO8z3wKNLl' // Gigi - XBio };

// Get voice ID for agent function getVoiceId(agentId: string): string { return AGENT_VOICES[agentId] || AGENT_VOICES.mrf; }

// Generate speech from text export async function textToSpeech(text: string, agentId: string): Promise { const voiceId = getVoiceId(agentId);

const audio = await elevenLabs.textToSpeech.convert(voiceId, { text, model_id: 'eleven_multilingual_v2', voice_settings: { stability: 0.5, similarity_boost: 0.75, style: 0.5, use_speaker_boost: true } });

// Convert stream to buffer const chunks: Buffer[] = []; for await (const chunk of audio) { chunks.push(Buffer.from(chunk)); }

return Buffer.concat(chunks); }

// Generate speech with streaming export async function textToSpeechStream(text: string, agentId: string): Promise<AsyncIterable> { const voiceId = getVoiceId(agentId);

return elevenLabs.textToSpeech.convertAsStream(voiceId, { text, model_id: 'eleven_multilingual_v2', voice_settings: { stability: 0.5, similarity_boost: 0.75 } }); }

========== ========== 4. API ROUTES (routes.ts) ========== ==========

/** * ARC Platform API Routes * Main entry point for all platform endpoints */

import { Router } from 'express'; import { db } from './db'; import { users, conversations, messages, agentStates, sectorMetrics } from '../shared/schema'; import { eq, desc, and } from 'drizzle-orm'; import { ALL_AGENTS, getAgent, routeMessage, getAgentsBySector } from './arc/hierarchy_system'; import { generateAgentResponse, generateCEOSummary } from './arc/openai_service'; import { textToSpeech } from './arc/voice_service'; import { requireAuth } from './middleware/auth';

const router = Router();

// ========== AGENTS ==========

// GET /api/agents - List all agents router.get('/agents', (req, res) => { res.json(ALL_AGENTS); });

// GET /api/agents/:id - Get specific agent router.get('/agents/:id', (req, res) => { const agent = getAgent(req.params.id); if (!agent) { return res.status(404).json({ error: 'Agent not found' }); } res.json(agent); });

// GET /api/agents/sector/:sector - Get agents by sector router.get('/agents/sector/:sector', (req, res) => { const agents = getAgentsBySector(req.params.sector as any); res.json(agents); });

// ============ CHAT ============

// POST /api/chat - Send message to agent router.post('/chat', requireAuth, async (req, res) => { try { const { message, agentId, conversationId } = req.body; const userId = req.user!.id;

```
// Get or determine agent
let agent = agentId ? getAgent(agentId) : routeMessage(message);
if (!agent) {
  return res.status(400).json({ error: 'Invalid agent' });
}

// Get conversation history
let history: { role: string; content: string }[] = [];
if (conversationId) {
  const msgs = await db.query.messages.findMany({
    where: eq(messages.conversationId, conversationId),
    orderBy: desc(messages.createdAt),
    limit: 10
  });
  history = msgs.reverse().map(m => ({
    role: m.role,
    content: m.content
  }));
}

// Generate response
const response = await generateAgentResponse(agent, message,
history);

// Store messages
if (conversationId) {
  await db.insert(messages).values([
    { conversationId, role: 'user', content: message },
    { conversationId, role: 'assistant', content: response, agentId:
agent.id }
  ]);
}

res.json({
  agent: {
    id: agent.id,
    name: agent.name,
    emoji: agent.emoji
  },
  message: response
});
```

} catch (error) { console.error('Chat error:', error); res.status(500).json({ error: 'Failed to process message' }); } });

// ============ VOICE ============

// POST /api/voice/synthesize - Text to speech
router.post('/voice/synthesize', requireAuth, async (req, res) => { try { const { text, agentId } = req.body;

```
const audioBuffer = await textToSpeech(text, agentId || 'mrf');

res.set({
  'Content-Type': 'audio/mpeg',
  'Content-Length': audioBuffer.length
});
res.send(audioBuffer);
```

} catch (error) { console.error('Voice synthesis error:', error); res.status(500).json({ error: 'Failed to synthesize voice' }); } });

// ============ DASHBOARD ============

// GET /api/dashboard/summary - CEO Dashboard summary
router.get('/dashboard/summary', requireAuth, async (req, res) => { try { const userId = req.user!.id;

```
// Get sector metrics
const metrics = await db.query.sectorMetrics.findMany({
  where: eq(sectorMetrics.userId, userId)
});

// Get recent conversations
const recentConvos = await db.query.conversations.findMany({
  where: eq(conversations.userId, userId),
  orderBy: desc(conversations.updatedAt),
  limit: 5
});

// Generate CEO summary
const sectorReports: Record<string, string> = {};
for (const metric of metrics) {
  sectorReports[metric.sector] = `Performance: ${metric.score}%,
Active tasks: ${metric.activeTasks}`;
}

const summary = Object.keys(sectorReports).length > 0
  ? await generateCEOSummary(sectorReports)
  : 'Welcome to ARC Virtual Office. All systems operational.';

res.json({
  summary,
  sectors: metrics,
  recentActivity: recentConvos,
  agentCount: ALL_AGENTS.length
});
```

} catch (error) { console.error('Dashboard error:', error); res.status(500).json({ error: 'Failed to load dashboard' }); } });

// ============ GROWTH ROADMAP ============

// GET /api/growth/phases - Get growth phases
router.get('/growth/phases', requireAuth, async (req, res) => { try { const userId = req.user!.id;

```
const phases = await db.query.growthPhases.findMany({
  where: eq(growthPhases.userId, userId),
```

```
  orderBy: growthPhases.order
});

res.json(phases);
```

} catch (error) { console.error(‘Growth phases error:’, error); res.status(500).json({ error: ‘Failed to load growth phases’ }); } });

// POST /api/growth/task/:id/complete - Complete a task router.post(‘/growth/task/:id/complete’, requireAuth, async (req, res) => { try { const taskId = req.params.id;

```
await db.update(growthTasks)
  .set({
    status: 'completed',
    completedAt: new Date()
  })
  .where(eq(growthTasks.id, taskId));

res.json({ success: true });
```

} catch (error) { console.error(‘Task completion error:’, error); res.status(500).json({ error: ‘Failed to complete task’ }); } });

export default router;

========== ========== 5. MRF DASHBOARD (MRFDashboard.tsx) ========== ==========

/** * MRF Dashboard - CEO Command Center * Central control panel for the 31-agent hierarchy */

import React, { useState, useEffect } from ‘react’; import { useQuery } from ‘@tanstack/react-query’; import { Card, CardHeader, CardTitle, CardContent } from ‘@/components/ui/card’; import { Badge } from ‘@/components/ui/badge’; import { Button } from ‘@/components/ui/button’; import { Crown, Shield, Wallet, Scale, Home, FlaskConical, Leaf, Users, MessageSquare, TrendingUp, Activity } from ‘lucide-react’;

interface SectorStatus { id: string; name: string; emoji: string; icon: any; score: number; activeTasks: number; color: string; }

export default function MRFDashboard() { const [selectedSector, setSelectedSector] = useState<string | null>(null);

const sectors: SectorStatus[] = [{ id: ‘security’, name: ‘الأمن’, emoji: ‘’, icon: Shield, score: 92, activeTasks: 3, color: ‘from-red-500 to-red-700’ }, { id: ‘finance’, name: ‘المالية’, emoji: ‘’, icon: Wallet, score: 88, activeTasks: 5, color: ‘from-green-500 to-green-700’ }, { id: ‘legal’, name: ‘القانون’, emoji: ‘⚖’, icon: Scale, score: 95, activeTasks: 2, color: ‘from-blue-500 to-blue-700’ }, { id: ‘life’, name: ‘الحياة’, emoji: ‘’, icon: Home, score: 78, activeTasks: 7, color: ‘from-purple-500 to-purple-700’ }, { id: ‘research’, name: ‘البحث والتطوير’, emoji: ‘’, icon: FlaskConical, score: 85, activeTasks: 4, color: ‘from-orange-500 to-orange-700’ }, { id: ‘xbio’, name: ‘البيئة’, emoji: ‘’, icon: Leaf, score: 90, activeTasks: 2, color: ‘from-teal-500 to-teal-700’ }];

const { data: summary } = useQuery({ queryKey: [‘dashboard-summary’], queryFn: async () => { const res = await fetch(‘/api/dashboard/summary’); return res.json(); }, refetchInterval: 30000 });

return (

```
{/* CEO Header */}
<"div className="mb-8>
<"div className="flex items-center gap-4>
div className="p-4 bg-gradient-to-br from-yellow-500 to->
<"yellow-700 rounded-2xl
</ "Crown className="h-10 w-10 text-white>
<div/>
<div>
لوحة تحكمh1 className="text-3xl font-bold text-white>
<Mr.F</h1
المكتب الافتراضي • 31 وكيل<"p className="text-gray-400>
<p/>
<div/>
<"Badge className="mr-auto bg-green-600 text-white px-4 py-2>
</ "Activity className="h-4 w-4 ml-2 inline>
ظام يعمل
<Badge/>
<div/>
<div/>

{/* Stats Overview */}
<"div className="grid grid-cols-4 gap-4 mb-8>
<"Card className="bg-gray-800 border-gray-700>
<"CardContent className="p-4 text-center>
</ "Users className="h-8 w-8 text-blue-400 mx-auto mb-2>
<div className="text-3xl font-bold text-white">31</div>
<div/>وكيل ذكي<"div className="text-gray-400 text-sm>
<CardContent/>
<Card/>
<"Card className="bg-gray-800 border-gray-700>
<"CardContent className="p-4 text-center>
MessageSquare className="h-8 w-8 text-green-400 mx-auto mb->
</ "2
<div className="text-3xl font-bold text-white">6</div>
<div/>قطاعات<"div className="text-gray-400 text-sm>
<CardContent/>
<Card/>
<"Card className="bg-gray-800 border-gray-700>
<"CardContent className="p-4 text-center>
TrendingUp className="h-8 w-8 text-yellow-400 mx-auto mb-2">
</
<div className="text-3xl font-bold text-white">88%</div>
<div/>الأداء العام<"div className="text-gray-400 text-sm>
<CardContent/>
<Card/>
<"Card className="bg-gray-800 border-gray-700>
<"CardContent className="p-4 text-center>
Activity className="h-8 w-8 text-purple-400 mx-auto mb-2">
</
<div className="text-3xl font-bold text-white">23</div>
<div/>مهمة نشطة<"div className="text-gray-400 text-sm>
<CardContent/>
<Card/>
<div/>
```

```
                                        {/* Sectors Grid */}
            <h2 className="text-xl font-bold text-white mb-4">حالة
                                                          </h2>
    <div className="grid grid-cols-2 md:grid-cols-3 gap-4 mb-8">
                                    {(sectors.map((sector) =>
                                                    <Card
                                                key={sector.id}
    className={`bg-gradient-to-br ${sector.color} border-0
                                                  cursor-pointer
        transition-all hover:scale-105 hover:shadow-xl`}
              onClick={() => setSelectedSector(sector.id)}>
                                                        <CardContent className="p-6">
  <div className="flex items-center justify-between mb-4">
          <div className="text-4xl">{sector.emoji}</div>
              <Badge className="bg-white/20 text-white">
                                {sector.activeTasks} مهام
                                            </Badge>
                                              </div>
      <h3 className="text-xl font-bold text-white mb-2">
                                              {sector.name}</h3>
               <div className="flex items-center gap-2">
   <div className="flex-1 bg-white/20 rounded-full h-2">
                                              <div
"className="bg-white rounded-full h-2 transition-all
              style={{ width: `${sector.score}%` }}
                                              />
                                              </div>
  <span className="text-white font-bold">{sector.score}%</span>
                                                    </div>
                                            </CardContent>
                                                </Card>
                                                    ))}
                                               </div>

                                        {/* CEO Summary */}
                                    {summary && summary?.summary) &&
                <Card className="bg-gray-800 border-gray-700">
                                              <CardHeader>
  <CardTitle className="text-white flex items-center gap-2">
            <Crown className="h-5 w-5 text-yellow-500" />
                                              ملخص Mr.F
                                              </CardTitle>
                                               </CardHeader>
                                               <CardContent>
              <p className="text-gray-300 leading-relaxed">
                                              {summary.summary}</p>
                                               </CardContent>
                                                </Card>
                                                    )}
                                               </div>

                                                    );}
```

==========
.6 ==========
(VIRTUAL OFFICE (VirtualOffice.tsx
==========
==========

```
Virtual Office - Agent Interaction Hub * Chat and voice * **/
/* interaction with 31 AI agents

import React, { useState, useRef } from 'react'; import { useQuery,
useMutation } from '@tanstack/react-query'; import { Card,
CardHeader, CardTitle, CardContent } from '@/components/ui/card';
import { Input } from '@/components/ui/input'; import { Button }
from '@/components/ui/button'; import { ScrollArea } from
'@/components/ui/scroll-area'; import { Send, Mic, Volume2, VolumeX
;'} from 'lucide-react

interface Agent { id: string; name: string; emoji: string; role: string;
{ ;sector: string

interface Message { id: string; role: 'user' | 'assistant'; content: string;
{ ;agentId?: string; timestamp: Date

export default function VirtualOffice() { const [messages,
setMessages] = useState<Message[]>([]); const [input, setInput] =
useState(''); const [selectedAgent, setSelectedAgent] =
useState<Agent | null>(null); const [isVoiceEnabled,
setIsVoiceEnabled] = useState(false); const [isRecording,
;(setIsRecording] = useState(false); const audioRef = useRef(null

Fetch agents const { data: agents } = useQuery<Agent[]>({ //
queryKey: ['agents'], queryFn: async () => { const res = await
;({ { ;()fetch('/api/agents'); return res.json

Send message mutation const sendMessage = useMutation({ //
mutationFn: async (message: string) => { const res = await
fetch('/api/chat', { method: 'POST', headers: { 'Content-Type':
'application/json' }, body: JSON.stringify({ message, agentId:
selectedAgent?.id }) }); return res.json(); }, onSuccess: async (data)
=> { const assistantMessage: Message = { id: Date.now().toString(),
role: 'assistant', content: data.message, agentId: data.agent.id,
timestamp: new Date() }; setMessages(prev => [...prev,
;([assistantMessage

                                    Voice synthesis if enabled //
                                          } (if (isVoiceEnabled
                        ;(await playVoice(data.message, data.agent.id
                                                                {
                                                                  {

                                                                ;({

Play voice const playVoice = async (text: string, agentId: string) => //
{ try { const res = await fetch('/api/voice/synthesize', { method:
'POST', headers: { 'Content-Type': 'application/json' }, body:
JSON.stringify({ text, agentId }) }); const audioBlob = await
;(res.blob(); const audioUrl = URL.createObjectURL(audioBlob

                                              } (if (audioRef.current
                                  ;audioRef.current.src = audioUrl
                                          ;()audioRef.current.play
                                                                {
                                                  } (catch (error {
                                ;(console.error('Voice playback error:', error
                                                                  {

                                                                ;{
```

```
;Handle send const handleSend = () => { if (!input.trim()) return //

                                        } = const userMessage: Message
                                          ,()id: Date.now().toString
                                                      ,'role: 'user
                                                    ,content: input
                                                 ()timestamp: new Date
                                                                    ;{
                                  ;([setMessages(prev => [...prev, userMessage
                                                   ;(sendMessage.mutate(input
                                                               ;('')setInput

                                                                        ;}

Get agent display info const getAgentInfo = (agentId?: string) => { // emoji: '' }; const agent ,'النظام' :if (!agentId || !agents) return { name ,'النظام' := agents.find(a => a.id === agentId); return agent || { name :emoji: '' ;{ ;{

                                                                ) return

                                                {/* Agents Sidebar */}
  <"div className="w-64 bg-gray-800 border-l border-gray-700 p-4>
<h2/>الوكلاء <"h2 className="text-lg font-bold text-white mb-4>
                  <"[(ScrollArea className="h-[calc(100vh-100px>
                                       ) <= agents?.map(agent}
                                                    Button>
                                                {key={agent.id
    variant={selectedAgent?.id === agent.id ? 'default' :
                                                      {''ghost
         "className="w-full justify-start mb-1 text-right
                  {((onClick={() => setSelectedAgent(agent
                                                     <
             <span className="ml-2">{agent.emoji}</span>
                                               {agent.name}
                                                    <Button/>
                                                           {((
                                                    <ScrollArea/>
                                                            <div/>

                                                 {/* Chat Area */}
                              <"div className="flex-1 flex flex-col>
                                                  {/* Header */}
  div className="bg-gray-800 border-b border-gray-700 p-4 flex>
                                         <"items-center justify-between
                      <"div className="flex items-center gap-3>
   span className="text-2xl">{selectedAgent?.emoji || ''}>
                                                              <</span
                                                             <div>
            <"h2 className="text-lg font-bold text-white>
             {'المكتب الافتراضي' || selectedAgent?.name}
                                                           <h2/>
                    <"p className="text-gray-400 text-sm>
           {'تحدث مع أي وكيل' || selectedAgent?.sector}
                                                           <p/>
                                                           <div/>
                                                             <div/>
                                                            Button>
                                                "variant="ghost
                                                    "size="icon
         {((onClick={() => setIsVoiceEnabled(!isVoiceEnabled
```

```
                                                        <
                                        ) ? isVoiceEnabled}
            </ "Volume2 className="h-5 w-5 text-green-400>
                                                    ) : (
             </ "VolumeX className="h-5 w-5 text-gray-400>
                                                       {(
                                                  <Button/>
                                                       <div/>

                                               {/* Messages */}
                                  <"ScrollArea className="flex-1 p-4>
                                             } <= messages.map(msg}
                ;(const agentInfo = getAgentInfo(msg.agentId
                                                  ) return
                                                    div>
                                           {key={msg.id
 className={`mb-4 flex ${msg.role === 'user' ? 'justify-
                                          {`{'end' : 'justify-start
                                                      <
                                                  div>
            }$ className={`max-w-[70%] rounded-2xl p-4
                                      'msg.role === 'user
                        'bg-blue-600 text-white' ?
                        'bg-gray-700 text-white' :
                                                {`{
                                                   <
                        ) && 'msg.role === 'assistant}
div className="flex items-center gap-2 mb-2 text-sm>
                                                    <"text-gray-300
                     <span>{agentInfo.emoji}</span>
                      <span>{agentInfo.name}</span>
                                             <div/>
                                                   {(
                                  <p>{msg.content}</p>
                                                <div/>
                                                  <div/>
                                                        ;(
                                                         {({
                                                 <ScrollArea/>

                                                 {/* Input */}
      <"div className="bg-gray-800 border-t border-gray-700 p-4>
                                    <"div className="flex gap-2>
                                                    Input>
                                           {value={input
               {(onChange={(e) => setInput(e.target.value
                                 "...اكتب رسالتك"=placeholder
 "className="flex-1 bg-gray-700 border-gray-600 text-white
     {()onKeyPress={(e) => e.key === 'Enter' && handleSend
                                                        </
                                                  Button>
                                          "variant="ghost
                                              "size="icon
             {(onClick={() => setIsRecording(!isRecording
                                                       <
 Mic className={`h-5 w-5 ${isRecording ? 'text-red-500' :
                                               </ {`{''text-gray-400
                                                  <Button/>
                      Button onClick={handleSend} disabled=>
                                           <{{sendMessage.isPending
```

```
                    </ "Send className="h-5 w-5>
                                    <Button/>
                                         <div/>
                                           <div/>
                                             <div/>

                                {/* Hidden audio element */}
                        </ "audio ref={audioRef} className="hidden>
                                                  <div/>

                                                     );{

==========================================
     نهاية الملف ==========================================
==========================================
     ==========================================
```


معيد المراجعة: SA 1020258841