Clone Hub IP Filing

==========
========== Clone Hub - Source Code Extract للتقديم لهيئة الملكية الفكرية السعودية (SAIP)
==========
==========

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

==========
========== وصف المنتج
==========
==========

Clone Hub هي منصة متكاملة لإدارة الموارد البشرية والتوأم الرقمي، تمكّن المؤسسات من: - إنشاء نسخة رقمية ذكية (Digital Twin) لكل موظف - إدارة الموظفين والصلاحيات - أتمتة المهام الإدارية والروتينية - التكامل مع أنظمة IoT (XBio Sentinel) (HR/Admin)

المكونات الرئيسية: - نظام الإدارة (Admin Panel) - نظام الاستنساخ الرقمي (Digital Twin) - نظام التكاملات والـ APIs HR الأساسي

==========
========== 1. DATABASE SCHEMA (schema.ts)
==========
==========

/** * Clone Hub Database Schema * Complete schema for HR and Digital Twin system */

import { pgTable, text, varchar, integer, boolean, timestamp, jsonb, uuid } from 'drizzle-orm/pg-core'; import { relations } from 'drizzle-orm';

// ========== USER PROFILES ==========

export const userProfiles = pgTable('user_profiles', { id: uuid('id').primaryKey().defaultRandom(), username: varchar('username', { length: 50 }).notNull().unique(), email: varchar('email', { length: 255 }).notNull().unique(), phoneNumber: varchar('phone_number', { length: 20 }), passwordHash: text('password_hash').notNull(),

// Personal Info personalInfo: jsonb('personal_info').$type<{ fullName?: string; arabicName?: string; dateOfBirth?: string; nationality?: string; gender?: string; bio?: string; skills?: string[]; languages?: string[]; timezone?: string; }>().default({}),

// Work Info workInfo: jsonb('work_info').$type<{ jobTitle?: string; department?: string; employeeId?: string; startDate?: string; manager?: string; workLocation?: string; workType?: 'onsite' | 'remote' | 'hybrid'; }>().default({}),

// Social Links socialInfo: jsonb('social_info').$type<{ linkedin?: string; twitter?: string; github?: string; portfolio?: string; telegram?: string; }>().default({}),

```
// Clone Settings cloneSettings: jsonb('clone_settings').$type<{
voiceModelId?: string; personalityProfile?: string; autoReplyEnabled?:
boolean; officeHours?: { start: string; end: string }[];
greetingMessage?: string; }>().default({}),

// Status isActive: boolean('is_active').default(true), isCloneEnabled:
boolean('is_clone_enabled').default(false), role: varchar('role', {
length: 20 }).default('employee'), // admin, manager, employee

createdAt: timestamp('created_at').defaultNow(), updatedAt:
timestamp('updated_at').defaultNow() });

// ============ USER FILES ============

export const userFiles = pgTable('user_files', { id:
uuid('id').primaryKey().defaultRandom(), userId:
uuid('user_id').notNull().references(() => userProfiles.id, { onDelete:
'cascade' }),

fileType: varchar('file_type', { length: 20 }).notNull(), // 'voice',
'photo', 'document', 'resume' fileName: varchar('file_name', { length:
255 }).notNull(), filePath: text('file_path').notNull(), fileSize:
integer('file_size'), // bytes mimeType: varchar('mime_type', { length:
100 }),

// Metadata metadata: jsonb('metadata').$type<{ duration?: number;
// for audio dimensions?: { w: number; h: number }; // for images
pageCount?: number; // for documents transcription?: string; // for
voice files }>().default({}),

isProcessed: boolean('is_processed').default(false), processedAt:
timestamp('processed_at'),

uploadedAt: timestamp('uploaded_at').defaultNow() });

// ============ IoT DEVICES ============

export const userIotDevices = pgTable('user_iot_devices', { id:
uuid('id').primaryKey().defaultRandom(), userId:
uuid('user_id').notNull().references(() => userProfiles.id, { onDelete:
'cascade' }),

deviceType: varchar('device_type', { length: 50 }).notNull(), //
'xbio_sentinel', 'personal_xbio', etc. deviceName:
varchar('device_name', { length: 100 }), deviceId: varchar('device_id',
{ length: 100 }).unique(),

deviceConfig: jsonb('device_config').$type<{ location?: string;
alertThresholds?: Record<string, number>; syncInterval?: number;
isPublic?: boolean; }>().default({}),

isActive: boolean('is_active').default(true), lastSyncAt:
timestamp('last_sync_at'),

createdAt: timestamp('created_at').defaultNow() });

// ============ INTEGRATIONS ============

export const userIntegrations = pgTable('user_integrations', { id:
uuid('id').primaryKey().defaultRandom(), userId:
uuid('user_id').notNull().references(() => userProfiles.id, { onDelete:
'cascade' }),
```

provider: varchar(‘provider’, { length: 50 }).notNull(), // ‘google’, ‘github’, ‘openai’, etc. providerUserId: varchar(‘provider_user_id’, { length: 255 }),

accessToken: text(‘access_token’), refreshToken: text(‘refresh_token’), tokenExpiresAt: timestamp(‘token_expires_at’),

scopes: text(‘scopes’).array(), metadata: jsonb(‘metadata’).default({}),

isActive: boolean(‘is_active’).default(true), lastUsedAt: timestamp(‘last_used_at’),

createdAt: timestamp(‘created_at’).defaultNow() });

// ============ LEAVE REQUESTS ============

export const leaveRequests = pgTable(‘leave_requests’, { id: uuid(‘id’).primaryKey().defaultRandom(), userId: uuid(‘user_id’).notNull().references(() => userProfiles.id, { onDelete: ‘cascade’ }),

leaveType: varchar(‘leave_type’, { length: 30 }).notNull(), // ‘annual’, ‘sick’, ‘emergency’, ‘maternity’ startDate: timestamp(‘start_date’).notNull(), endDate: timestamp(‘end_date’).notNull(), totalDays: integer(‘total_days’).notNull(),

reason: text(‘reason’), status: varchar(‘status’, { length: 20 }).default(‘pending’), // ‘pending’, ‘approved’, ‘rejected’

approvedBy: uuid(‘approved_by’).references(() => userProfiles.id), approvedAt: timestamp(‘approved_at’), rejectionReason: text(‘rejection_reason’),

createdAt: timestamp(‘created_at’).defaultNow() });

// ============ ATTENDANCE ============

export const attendance = pgTable(‘attendance’, { id: uuid(‘id’).primaryKey().defaultRandom(), userId: uuid(‘user_id’).notNull().references(() => userProfiles.id, { onDelete: ‘cascade’ }),

date: timestamp(‘date’).notNull(), checkIn: timestamp(‘check_in’), checkOut: timestamp(‘check_out’),

workHours: integer(‘work_hours’), // minutes breakMinutes: integer(‘break_minutes’), overtimeMinutes: integer(‘overtime_minutes’),

status: varchar(‘status’, { length: 20 }).default(‘present’), // ‘present’, ‘absent’, ‘late’, ‘half_day’ notes: text(‘notes’),

location: jsonb(‘location’).$type<{ checkInLocation?: { lat: number; lng: number }; checkOutLocation?: { lat: number; lng: number }; }>().default({}) });

// ============ PERFORMANCE REVIEWS ============

export const performanceReviews = pgTable(‘performance_reviews’, { id: uuid(‘id’).primaryKey().defaultRandom(), userId: uuid(‘user_id’).notNull().references(() => userProfiles.id, { onDelete: ‘cascade’ }), reviewerId: uuid(‘reviewer_id’).notNull().references(() => userProfiles.id),

reviewPeriod: varchar(‘review_period’, { length: 20 }).notNull(), // ‘Q1-2026’, ‘Annual-2025’

scores: jsonb(‘scores’).$type<{ productivity?: number; quality?: number; teamwork?: number; communication?: number; innovation?: number; attendance?: number; overall?: number; }>().default({}),

strengths: text(‘strengths’).array(), improvements: text(‘improvements’).array(), goals: text(‘goals’).array(), comments: text(‘comments’),

status: varchar(‘status’, { length: 20 }).default(‘draft’), // ‘draft’, ‘submitted’, ‘acknowledged’ acknowledgedAt: timestamp(‘acknowledged_at’),

createdAt: timestamp(‘created_at’).defaultNow() });

// ============ CLONE CONVERSATIONS ============

export const cloneConversations = pgTable(‘clone_conversations’, { id: uuid(‘id’).primaryKey().defaultRandom(), cloneOwnerId: uuid(‘clone_owner_id’).notNull().references(() => userProfiles.id), visitorId: uuid(‘visitor_id’).references(() => userProfiles.id), // null for anonymous

visitorName: varchar(‘visitor_name’, { length: 100 }), visitorEmail: varchar(‘visitor_email’, { length: 255 }),

status: varchar(‘status’, { length: 20 }).default(‘active’), // ‘active’, ‘closed’, ‘escalated’

summary: text(‘summary’), sentiment: varchar(‘sentiment’, { length: 20 }), // ‘positive’, ‘neutral’, ‘negative’

escalatedTo: uuid(‘escalated_to’).references(() => userProfiles.id), escalatedAt: timestamp(‘escalated_at’),

createdAt: timestamp(‘created_at’).defaultNow(), closedAt: timestamp(‘closed_at’) });

// ============ CLONE MESSAGES ============

export const cloneMessages = pgTable(‘clone_messages’, { id: uuid(‘id’).primaryKey().defaultRandom(), conversationId: uuid(‘conversation_id’).notNull().references(() => cloneConversations.id, { onDelete: ‘cascade’ }),

role: varchar(‘role’, { length: 20 }).notNull(), // ‘visitor’, ‘clone’, ‘human’ content: text(‘content’).notNull(),

// If clone responded aiModel: varchar(‘ai_model’, { length: 50 }), tokensUsed: integer(‘tokens_used’), responseTime: integer(‘response_time’), // milliseconds

createdAt: timestamp(‘created_at’).defaultNow() });

// ============ KNOWLEDGE BASE ============

export const knowledgeBase = pgTable(‘knowledge_base’, { id: uuid(‘id’).primaryKey().defaultRandom(), userId: uuid(‘user_id’).notNull().references(() => userProfiles.id, { onDelete: ‘cascade’ }),

title: varchar(‘title’, { length: 255 }).notNull(), content: text(‘content’).notNull(), category: varchar(‘category’, { length: 50 }), // ‘faq’, ‘policy’, ‘personal’, ‘work’

// Vector embedding for semantic search embedding: text('embedding'),

isPublic: boolean('is_public').default(false), isActive: boolean('is_active').default(true),

createdAt: timestamp('created_at').defaultNow(), updatedAt: timestamp('updated_at').defaultNow() });

// ============ RELATIONS ============

export const userProfilesRelations = relations(userProfiles, ({ many }) => ({ files: many(userFiles), devices: many(userIotDevices), integrations: many(userIntegrations), leaveRequests: many(leaveRequests), attendanceRecords: many(attendance), knowledgeItems: many(knowledgeBase) }));

== == 2. ADMIN ROUTES (admin-routes.ts) == ==

/** * Clone Hub Admin API Routes * HR and Admin panel endpoints */

import { Router } from 'express'; import { db } from '../db'; import { userProfiles, userFiles, leaveRequests, attendance, performanceReviews } from '../../shared/schema'; import { eq, desc, and, gte, lte, sql } from 'drizzle-orm'; import { requireAuth, requireRole } from '../middleware/auth'; import bcrypt from 'bcrypt';

const router = Router();

// ============ USER MANAGEMENT ============

// GET /api/admin/users - List all users (Admin only) router.get('/users', requireAuth, requireRole(['admin', 'manager']), async (req, res) => { try { const { page = 1, limit = 20, department, status } = req.query; const offset = (Number(page) - 1) * Number(limit);

```
const users = await db.query.userProfiles.findMany({
  limit: Number(limit),
  offset,
  orderBy: desc(userProfiles.createdAt),
  with: {
    files: {
      where: eq(userFiles.fileType, 'photo'),
      limit: 1
    }
  }
});

const total = await db.select({ count: sql`count(*)` })
  .from(userProfiles);

res.json({
  users,
  pagination: {
    page: Number(page),
    limit: Number(limit),
    total: Number(total[0].count),
    pages: Math.ceil(Number(total[0].count) / Number(limit))
  }
```

```
});
```

} catch (error) { console.error('Error fetching users:', error); res.status(500).json({ error: 'Failed to fetch users' }); } });

// POST /api/admin/users - Create new user router.post('/users', requireAuth, requireRole(['admin']), async (req, res) => { try { const { username, email, password, role, personalInfo, workInfo } = req.body;

```
// Check if exists
const existing = await db.query.userProfiles.findFirst({
  where: eq(userProfiles.email, email)
});

if (existing) {
  return res.status(400).json({ error: 'Email already exists' });
}

// Hash password
const passwordHash = await bcrypt.hash(password, 12);

// Create user
const user = await db.insert(userProfiles).values({
  username,
  email,
  passwordHash,
  role: role || 'employee',
  personalInfo: personalInfo || {},
  workInfo: workInfo || {}
}).returning();

res.json({ success: true, user: user[0] });
```

} catch (error) { console.error('Error creating user:', error); res.status(500).json({ error: 'Failed to create user' }); } });

// PUT /api/admin/users/:id - Update user router.put('/users/:id', requireAuth, requireRole(['admin', 'manager']), async (req, res) => { try { const { id } = req.params; const updates = req.body;

```
// Remove sensitive fields from updates
delete updates.passwordHash;
delete updates.id;

const user = await db.update(userProfiles)
  .set({ ...updates, updatedAt: new Date() })
  .where(eq(userProfiles.id, id))
  .returning();

res.json({ success: true, user: user[0] });
```

} catch (error) { console.error('Error updating user:', error); res.status(500).json({ error: 'Failed to update user' }); } });

// DELETE /api/admin/users/:id - Deactivate user router.delete('/users/:id', requireAuth, requireRole(['admin']), async (req, res) => { try { const { id } = req.params;

```
await db.update(userProfiles)
  .set({ isActive: false, updatedAt: new Date() })
  .where(eq(userProfiles.id, id));

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

} catch (error) { console.error('Error deactivating user:', error); res.status(500).json({ error: 'Failed to deactivate user' }); } });

// ============ LEAVE MANAGEMENT ============

// GET /api/admin/leaves - List leave requests router.get('/leaves', requireAuth, requireRole(['admin', 'manager']), async (req, res) => { try { const { status, startDate, endDate } = req.query;

```
let query = db.query.leaveRequests.findMany({
  with: {
    user: true,
    approver: true
  },
  orderBy: desc(leaveRequests.createdAt)
});

const leaves = await query;

res.json(leaves);
```

} catch (error) { console.error('Error fetching leaves:', error); res.status(500).json({ error: 'Failed to fetch leave requests' }); } });

// POST /api/admin/leaves/:id/approve - Approve leave router.post('/leaves/:id/approve', requireAuth, requireRole(['admin', 'manager']), async (req, res) => { try { const { id } = req.params; const approverId = req.user!.id;

```
await db.update(leaveRequests)
  .set({
    status: 'approved',
    approvedBy: approverId,
    approvedAt: new Date()
  })
  .where(eq(leaveRequests.id, id));

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

} catch (error) { console.error('Error approving leave:', error); res.status(500).json({ error: 'Failed to approve leave' }); } });

// POST /api/admin/leaves/:id/reject - Reject leave router.post('/leaves/:id/reject', requireAuth, requireRole(['admin', 'manager']), async (req, res) => { try { const { id } = req.params; const { reason } = req.body; const approverId = req.user!.id;

```
await db.update(leaveRequests)
  .set({
    status: 'rejected',
    approvedBy: approverId,
    approvedAt: new Date(),
    rejectionReason: reason
  })
  .where(eq(leaveRequests.id, id));

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

} catch (error) { console.error('Error rejecting leave:', error); res.status(500).json({ error: 'Failed to reject leave' }); } });

// ============ ATTENDANCE ============

// GET /api/admin/attendance - Get attendance records router.get('/attendance', requireAuth, requireRole(['admin', 'manager']), async (req, res) => { try { const { date, userId } = req.query;

```
const records = await db.query.attendance.findMany({
  with: { user: true },
  orderBy: desc(attendance.date),
  limit: 100
});

res.json(records);
```

} catch (error) { console.error('Error fetching attendance:', error); res.status(500).json({ error: 'Failed to fetch attendance' }); } });

// GET /api/admin/attendance/summary - Attendance summary router.get('/attendance/summary', requireAuth, requireRole(['admin', 'manager']), async (req, res) => { try { const today = new Date(); today.setHours(0, 0, 0, 0);

```
const todayRecords = await db.query.attendance.findMany({
  where: gte(attendance.date, today)
});

const summary = {
  present: todayRecords.filter(r => r.status === 'present').length,
  absent: todayRecords.filter(r => r.status === 'absent').length,
  late: todayRecords.filter(r => r.status === 'late').length,
  onLeave: 0 // Calculate from leave requests
};

res.json(summary);
```

} catch (error) { console.error('Error fetching summary:', error); res.status(500).json({ error: 'Failed to fetch summary' }); } });

// ============ PERFORMANCE ============

// GET /api/admin/reviews - Get performance reviews router.get('/reviews', requireAuth, requireRole(['admin', 'manager']), async (req, res) => { try { const reviews = await db.query.performanceReviews.findMany({ with: { user: true, reviewer: true }, orderBy: desc(performanceReviews.createdAt) });

```
res.json(reviews);
```

} catch (error) { console.error('Error fetching reviews:', error); res.status(500).json({ error: 'Failed to fetch reviews' }); } });

// POST /api/admin/reviews - Create performance review router.post('/reviews', requireAuth, requireRole(['admin', 'manager']), async (req, res) => { try { const reviewerId = req.user!.id; const { userId, reviewPeriod, scores, strengths, improvements, goals, comments } = req.body;

```
const review = await db.insert(performanceReviews).values({
  userId,
  reviewerId,
  reviewPeriod,
  scores,
  strengths,
  improvements,
  goals,
```

```
  comments
}).returning();

res.json({ success: true, review: review[0] });
```

} catch (error) { console.error('Error creating review:', error); res.status(500).json({ error: 'Failed to create review' }); } });

// ============ ANALYTICS ============

// GET /api/admin/analytics - HR analytics dashboard router.get('/analytics', requireAuth, requireRole(['admin']), async (req, res) => { try { const totalUsers = await db.select({ count: sql`count(*)` }) .from(userProfiles) .where(eq(userProfiles.isActive, true));

```
const byDepartment = await db.select({
  department: sql`work_info->>'department'`,
  count: sql`count(*)`
})
  .from(userProfiles)
  .where(eq(userProfiles.isActive, true))
  .groupBy(sql`work_info->>'department'`);

const pendingLeaves = await db.select({ count: sql`count(*)` })
  .from(leaveRequests)
  .where(eq(leaveRequests.status, 'pending'));

res.json({
  totalEmployees: Number(totalUsers[0].count),
  byDepartment,
  pendingLeaveRequests: Number(pendingLeaves[0].count),
  clonesEnabled: 0 // Calculate
});
```

} catch (error) { console.error('Error fetching analytics:', error); res.status(500).json({ error: 'Failed to fetch analytics' }); } });

export default router;

========== ========== 3. CLONING ROUTES (cloning-routes.ts) ========== ==========

/** * Clone Hub - Digital Twin API Routes * Endpoints for digital clone creation and interaction */

import { Router } from 'express'; import { db } from '../db'; import { userProfiles, userFiles, knowledgeBase, cloneConversations, cloneMessages } from '../../shared/schema'; import { eq, desc, and } from 'drizzle-orm'; import { requireAuth } from '../middleware/auth'; import OpenAI from 'openai'; import multer from 'multer'; import path from 'path';

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

// File upload config const storage = multer.diskStorage({ destination: './uploads/clones', filename: (req, file, cb) => { const uniqueName = `${Date.now()}-${Math.random().toString(36).substring(7)}${path.extname(file.originalname)}`; cb(null, uniqueName); } });

const upload = multer({ storage, limits: { fileSize: 50 * 1024 * 1024 }, // 50MB fileFilter: (req, file, cb) => { const allowedTypes = [‘audio/mpeg’, ‘audio/wav’, ‘audio/ogg’, ‘audio/webm’, ‘image/jpeg’, ‘image/png’, ‘image/gif’, ‘image/webp’, ‘application/pdf’, ‘text/plain’, ‘application/msword’, ‘application/vnd.openxmlformats-officedocument.wordprocessingml.document’]; if (allowedTypes.includes(file.mimetype)) { cb(null, true); } else { cb(new Error(‘Invalid file type’)); } } });

// ============ CLONE SETUP ============

// POST /api/clone/setup - Initialize clone for user router.post(‘/setup’, requireAuth, async (req, res) => { try { const userId = req.user!.id; const { personalityProfile, greetingMessage, officeHours } = req.body;

```
await db.update(userProfiles)
  .set({
    isCloneEnabled: true,
    cloneSettings: {
      personalityProfile,
      greetingMessage,
      officeHours,
      autoReplyEnabled: true
    },
    updatedAt: new Date()
  })
  .where(eq(userProfiles.id, userId));

res.json({ success: true, message: 'Clone setup complete' });
```

} catch (error) { console.error(‘Clone setup error:’, error); res.status(500).json({ error: ‘Failed to setup clone’ }); } });

// POST /api/clone/upload - Upload files for clone training router.post(‘/upload’, requireAuth, upload.array(‘files’, 10), async (req, res) => { try { const userId = req.user!.id; const files = req.files as Express.Multer.File[]; const { fileType } = req.body; // ‘voice’, ‘photo’, ‘document’

```
const insertedFiles = [];

for (const file of files) {
  const fileRecord = await db.insert(userFiles).values({
    userId,
    fileType,
    fileName: file.originalname,
    filePath: file.path,
    fileSize: file.size,
    mimeType: file.mimetype
  }).returning();

  insertedFiles.push(fileRecord[0]);
}

// Trigger processing (voice training, OCR, etc.)
processFiles(insertedFiles);

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

} catch (error) { console.error(‘Upload error:’, error); res.status(500).json({ error: ‘Failed to upload files’ }); } });

// POST /api/clone/knowledge - Add knowledge base entry
router.post('/knowledge', requireAuth, async (req, res) => { try {
const userId = req.user!.id; const { title, content, category, isPublic }
= req.body;

```
// Generate embedding for semantic search
const embeddingResponse = await openai.embeddings.create({
  model: 'text-embedding-3-small',
  input: `${title}\n${content}`
});
const embedding =
JSON.stringify(embeddingResponse.data[0].embedding);

const entry = await db.insert(knowledgeBase).values({
  userId,
  title,
  content,
  category,
  isPublic,
  embedding
}).returning();

res.json({ success: true, entry: entry[0] });
```

} catch (error) { console.error('Knowledge add error:', error);
res.status(500).json({ error: 'Failed to add knowledge' }); } });

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

```
const entries = await db.query.knowledgeBase.findMany({
  where: and(
    eq(knowledgeBase.userId, userId),
    eq(knowledgeBase.isActive, true)
  ),
  orderBy: desc(knowledgeBase.createdAt)
});

res.json(entries);
```

} catch (error) { console.error('Knowledge fetch error:', error);
res.status(500).json({ error: 'Failed to fetch knowledge' }); } });

// ============ CLONE INTERACTION ============

// POST /api/clone/:userId/chat - Chat with someone's clone
router.post('/:userId/chat', async (req, res) => { try { const { userId }
= req.params; const { message, conversationId, visitorName,
visitorEmail } = req.body;

```
// Get clone owner
const owner = await db.query.userProfiles.findFirst({
  where: and(
    eq(userProfiles.id, userId),
    eq(userProfiles.isCloneEnabled, true)
  )
});

if (!owner) {
  return res.status(404).json({ error: 'Clone not found or not
enabled' });
}
```

```
// Get or create conversation
let convoId = conversationId;
if (!convoId) {
  const convo = await db.insert(cloneConversations).values({
    cloneOwnerId: userId,
    visitorName,
    visitorEmail
  }).returning();
  convoId = convo[0].id;
}

// Store visitor message
await db.insert(cloneMessages).values({
  conversationId: convoId,
  role: 'visitor',
  content: message
});

// Get knowledge base for context
const knowledge = await db.query.knowledgeBase.findMany({
  where: and(
    eq(knowledgeBase.userId, userId),
    eq(knowledgeBase.isActive, true)
  ),
  limit: 10
});

// Get conversation history
const history = await db.query.cloneMessages.findMany({
  where: eq(cloneMessages.conversationId, convoId),
  orderBy: desc(cloneMessages.createdAt),
  limit: 10
});

// Build clone prompt
const systemPrompt = buildClonePrompt(owner, knowledge);

// Generate response
const startTime = Date.now();
const completion = await openai.chat.completions.create({
  model: 'gpt-4',
  messages: [
    { role: 'system', content: systemPrompt },
    ...history.reverse().map(m => ({
      role: m.role === 'visitor' ? 'user' as const : 'assistant' as
const,
      content: m.content
    })),
    { role: 'user', content: message }
  ],
  temperature: 0.7,
  max_tokens: 500
});

const responseTime = Date.now() - startTime;
const response = completion.choices[0]?.message?.content || 'عذراً،
لم أتمكن من الرد.';

// Store clone response
await db.insert(cloneMessages).values({
  conversationId: convoId,
  role: 'clone',
  content: response,
```

```
  aiModel: 'gpt-4',
  tokensUsed: completion.usage?.total_tokens,
  responseTime
});

res.json({
  conversationId: convoId,
  message: response,
  owner: {
    name: owner.personalInfo?.fullName || owner.username,
    avatar: null // Would come from files
  }
});
```

} catch (error) { console.error('Clone chat error:', error); res.status(500).json({ error: 'Failed to chat with clone' }); } });

// GET /api/clone/:userId/conversations - Get clone conversations (owner only) router.get('/:userId/conversations', requireAuth, async (req, res) => { try { const { userId } = req.params;

```
// Verify owner
if (req.user!.id !== userId && req.user!.role !== 'admin') {
  return res.status(403).json({ error: 'Unauthorized' });
}

const conversations = await db.query.cloneConversations.findMany({
  where: eq(cloneConversations.cloneOwnerId, userId),
  orderBy: desc(cloneConversations.createdAt),
  with: {
    messages: {
      limit: 1,
      orderBy: desc(cloneMessages.createdAt)
    }
  }
});

res.json(conversations);
```

} catch (error) { console.error('Conversations fetch error:', error); res.status(500).json({ error: 'Failed to fetch conversations' }); } });

// POST /api/clone/:userId/escalate - Escalate to human router.post('/:userId/escalate', async (req, res) => { try { const { conversationId, reason } = req.body; const { userId } = req.params;

```
await db.update(cloneConversations)
  .set({
    status: 'escalated',
    escalatedTo: userId,
    escalatedAt: new Date()
  })
  .where(eq(cloneConversations.id, conversationId));

// Notify owner (email, push, etc.)
// notifyOwner(userId, conversationId, reason);

res.json({ success: true, message: 'Escalated to human' });
```

} catch (error) { console.error('Escalation error:', error); res.status(500).json({ error: 'Failed to escalate' }); } });

// ============ HELPER FUNCTIONS ============

function buildClonePrompt(owner: any, knowledge: any[]): string { const settings = owner.cloneSettings || {}; const personal = owner.personalInfo || {}; const work = owner.workInfo || {};

let prompt = `You are a digital clone/assistant representing ${personal.fullName || owner.username}.

About the person you represent: - Name: ${personal.fullName || owner.username} - Job: ${work.jobTitle || 'Not specified'} - Department: ${work.department || 'Not specified'} - Bio: ${personal.bio || 'Not provided'} - Skills: ${personal.skills?.join(',') || 'Not specified'}

Personality: ${settings.personalityProfile || 'Professional, helpful, and friendly'}

Greeting: ${settings.greetingMessage || مرحباً! أنا المساعد الرقمي لـ ${personal.fullName || owner.username}. كيف يمكنني مساعدتك؟}

Knowledge Base: `;

for (const item of knowledge) { prompt += \n[${item.category}] ${item.title}: ${item.content.substring(0, 200)}...; }

prompt += `

Guidelines: 1. Respond as if you ARE this person's digital assistant/clone 2. Use their name in third person when appropriate 3. If you don't know something, offer to escalate to the real person 4. Be helpful but set boundaries - you can't make commitments on their behalf 5. Respond in the same language as the visitor (Arabic or English) 6. If asked about availability, refer to office hours if set `;

return prompt; }

async function processFiles(files: any[]) { // Background processing for uploaded files for (const file of files) { if (file.fileType === 'voice') { // Transcribe and analyze voice console.log(Processing voice file: ${file.fileName}); } else if (file.fileType === 'document') { // OCR and extract text console.log(Processing document: ${file.fileName}); }

```
// Mark as processed
await db.update(userFiles)
  .set({ isProcessed: true, processedAt: new Date() })
  .where(eq(userFiles.id, file.id));
```

} }

export default router;

==========
========== 4. ADMIN DASHBOARD (AdminDashboard.tsx)
==========
==========

/** * Clone Hub Admin Dashboard * HR Management Interface */

import React, { useState } from 'react'; import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query'; import { Card, CardHeader, CardTitle, CardContent } from '@/components/ui/card'; import { Button } from '@/components/ui/button'; import { Input } from '@/components/ui/input'; import { Badge } from '@/components/ui/badge'; import { Tabs, TabsList, TabsTrigger,

TabsContent } from '@/components/ui/tabs'; import { Users, UserPlus, Calendar, Clock, BarChart3, CheckCircle, XCircle, AlertCircle } from 'lucide-react';

interface User { id: string; username: string; email: string; personalInfo: { fullName?: string }; workInfo: { department?: string; jobTitle?: string }; role: string; isActive: boolean; isCloneEnabled: boolean; }

interface LeaveRequest { id: string; user: User; leaveType: string; startDate: string; endDate: string; totalDays: number; reason: string; status: string; }

export default function AdminDashboard() { const [activeTab, setActiveTab] = useState('overview'); const queryClient = useQueryClient();

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

// Fetch users const { data: usersData } = useQuery({ queryKey: ['admin-users'], queryFn: async () => { const res = await fetch('/api/admin/users'); return res.json(); } });

// Fetch leave requests const { data: leaves } = useQuery<LeaveRequest[]>({ queryKey: ['admin-leaves'], queryFn: async () => { const res = await fetch('/api/admin/leaves'); return res.json(); } });

// Approve/Reject mutations const approveLeave = useMutation({ mutationFn: async (id: string) => { await fetch(`/api/admin/leaves/${id}/approve`, { method: 'POST' }); }, onSuccess: () => queryClient.invalidateQueries({ queryKey: ['admin-leaves'] }) });

const rejectLeave = useMutation({ mutationFn: async ({ id, reason }: { id: string; reason: string }) => { await fetch(`/api/admin/leaves/${id}/reject`, { method: 'POST', headers: { 'Content-Type': 'application/json' }, body: JSON.stringify({ reason }) }); }, onSuccess: () => queryClient.invalidateQueries({ queryKey: ['admin-leaves'] }) });

const pendingLeaves = leaves?.filter(l => l.status === 'pending') || [];

return (

```
{/* Header */}
<"div className="mb-8>
الوحةh1-className="text-3xl font-bold text-white">□ Clone Hub>
<h1/>
<"p className="text-gray-400إدارة الموارد البشرية والتوأم
<p/>
<div/>

{/* Stats Cards */}
<"div className="grid grid-cols-4 gap-4 mb-8>
Card className="bg-gradient-to-br from-blue-600 to-blue-800>
<"border-0
<"CardContent className="p-6>
</ "Users className="h-8 w-8 text-white mb-2>
div className="text-3xl font-bold text-white">>
<{analytics?.totalEmployees || 0}</div
```

```
       <div/>إجمالي الموظفين<"div className="text-blue-200>
                                                <CardContent/>
                                                     <Card/>
  Card className="bg-gradient-to-br from-green-600 to-green-800>
                                                      <"border-0
                                  <"CardContent className="p-6>
         </ "CheckCircle className="h-8 w-8 text-white mb-2>
             div className="text-3xl font-bold text-white">>
                                 <{analytics?.clonesEnabled || 0}</div
       <div/>التوائم النشطة<"div className="text-green-200>
                                                <CardContent/>
                                                     <Card/>
Card className="bg-gradient-to-br from-yellow-600 to-yellow-800>
                                                      <"border-0
                                  <"CardContent className="p-6>
         </ "AlertCircle className="h-8 w-8 text-white mb-2>
             div className="text-3xl font-bold text-white">>
                                           <{pendingLeaves.length}</div
        <div/>طلبات معلقة<"div className="text-yellow-200>
                                                <CardContent/>
                                                     <Card/>
Card className="bg-gradient-to-br from-purple-600 to-purple-800>
                                                      <"border-0
                                  <"CardContent className="p-6>
          </ "BarChart3 className="h-8 w-8 text-white mb-2>
    <div className="text-3xl font-bold text-white">94%</div>
        <div/>نسبة الحضور<"div className="text-purple-200>
                                                <CardContent/>
                                                     <Card/>
                                                        <div/>

                                                {/* Tabs */}
           <{Tabs value={activeTab} onValueChange={setActiveTab>
                          <"TabsList className="bg-gray-800 mb-6>
       <TabsTrigger/>نظرة عامة<"TabsTrigger value="overview>
       <TabsTrigger/>الموظفين<"TabsTrigger value="employees>
           <TabsTrigger/>الإجازات<"TabsTrigger value="leaves>
        <TabsTrigger/>الحضور<"TabsTrigger value="attendance>
    <TabsTrigger/>التوائم الرقمية<"TabsTrigger value="clones>
                                                    <TabsList/>

                                           {/* Employees Tab */}
                                  <"TabsContent value="employees>
              <"Card className="bg-gray-800 border-gray-700>
  CardHeader className="flex flex-row items-center justify->
                                                      <"between
                        CardTitle className="text-white>قائمة
                                                     <CardTitle/>
                                                 <Button>
                 </ "UserPlus className="h-4 w-4 ml-2>
                                                        إضافة موظف
                                                <Button/>
                                               <CardHeader/>
                                               <CardContent>
                                    <"table className="w-full>
                                                 <thead>
            <"tr className="text-gray-400 text-right>
                      <th/>الاسم<"th className="p-3>
                     <th/>القسم<"th className="p-3>
                    <th/>المنصب<"th className="p-3>
                    <th/>الحالة<"th className="p-3>
                    <th/>التوأم<"th className="p-3>
                   <th/>إجراءات<"th className="p-3>
```

```
                                                <tr/>
                                               <thead/>
                                                <tbody>
            ) <= (usersData?.users?.map((user: User}
  tr key={user.id} className="border-t border-gray->
                                                    <"700 text-white
  td className="p-3">{user.personalInfo?.fullName>
                                              <|| user.username}</td
 td className="p-3">{user.workInfo?.department ||>
                                                      <'-'}</td
td className="p-3">{user.workInfo?.jobTitle || '->
                                                        <'}</td
                              <"td className="p-3>
Badge className={user.isActive ? 'bg-green-600'>
                                                    <{': 'bg-red-600
           {'غير نشط' : 'نشط' ? user.isActive}
                                         <Badge/>
                                              <td/>
                              <"td className="p-3>
    Badge className={user.isCloneEnabled ? 'bg->
                                          <{'blue-600' : 'bg-gray-600
       {'معطل' : 'مفعل' ? user.isCloneEnabled}
                                         <Badge/>
                                              <td/>
                              <"td className="p-3>
<Button/>تعديل<"Button variant="ghost" size="sm>
                                              <td/>
                                                <tr/>
                                                    {((
                                               <tbody/>
                                                  <table/>
                                             <CardContent/>
                                                      <Card/>
                                                <TabsContent/>

                                             {/* Leaves Tab */}
                                    <"TabsContent value="leaves>
                <"Card className="bg-gray-800 border-gray-700>
                                               <CardHeader>
                        CardTitle className="text-white>إجازات
                                                       <CardTitle/>
                                              <CardHeader/>
                                              <CardContent>
                                ) ? pendingLeaves.length === 0}
          لا توجد إجازات معلقة<"p className="text-gray-400 text-center py-8>
                                                          <p/>
                                                  ) : (
                                <"div className="space-y-4>
                            ) <= (pendingLeaves.map((leave}
 div key={leave.id} className="bg-gray-700 rounded->
                                                      <"lg p-4
 div className="flex items-center justify-between>
                                                        <"mb-3
                                           <div>
        <"h4 className="text-white font-bold>
     || leave.user?.personalInfo?.fullName}
                                                      موظف'}
                                          <h4/>
        <"p className="text-gray-400 text-sm>
     {leave.leaveType} • {leave.totalDays} أيام
                                           <p/>
                                           <div/>
  <Badge/>معلق<"Badge className="bg-yellow-600>
```

```
                                        <div/>
       <"p className="text-gray-300 text-sm mb-3>
                                      new} من
                {('Date(leave.startDate).toLocaleDateString('ar-SA
                                      new} إلى
                  {('Date(leave.endDate).toLocaleDateString('ar-SA
                                        <p/>
       p className="text-gray-400 text-sm mb-4">>
                                            <{leave.reason}</p
                    <"div className="flex gap-2>
                                      Button>
                                  "size="sm
                    "className="bg-green-600
{(onClick={() => approveLeave.mutate(leave.id
                                         <
     </ "CheckCircle className="h-4 w-4 ml-1>
                                          موافقة
                                    <Button/>
                                      Button>
                                  "size="sm
                         "variant="destructive
      onClick={() => rejectLeave.mutate({ id:
                                 {({ 'تم الرفض' :leave.id, reason
                                         <
        </ "XCircle className="h-4 w-4 ml-1>
                                          رفض
                                    <Button/>
                                        <div/>
                                          <div/>
                                               {((
                                              <div/>
                                                    {(
                                          <CardContent/>
                                                   <Card/>
                                              <TabsContent/>

                                      {/* ...Other tabs */}
                                                 <Tabs/>
                                                    <div/>

                                                       { ;(
```

==========
.5 ==========
(CLONE PROFILE PAGE (CloneProfile.tsx
==========
==========

/* Clone Profile - Digital Twin Setup Interface * **/

import React, { useState } from ‘react’; import { useQuery, useMutation } from ‘@tanstack/react-query’; import { Card, CardHeader, CardTitle, CardContent } from ‘@/components/ui/card’; import { Button } from ‘@/components/ui/button’; import { Input } from ‘@/components/ui/input’; import { Textarea } from ‘@/components/ui/textarea’; import { Switch } from ‘@/components/ui/switch’; import { User, Mic, Image, FileText, ;’Upload, Brain, MessageSquare, Settings } from ‘lucide-react

export default function CloneProfile() { const [files, setFiles] = useState<File[]>([]); const [uploading, setUploading] = ;(useState(false

Upload files const uploadFiles = useMutation({ mutationFn: async //
({ files, type }: { files: File[], type: string }) => { setUploading(true);
const formData = new FormData(); files.forEach(f =>
;(formData.append(‘files’, f)); formData.append(‘fileType’, type

```
} ,'const res = await fetch('/api/clone/upload
,'method: 'POST
body: formData
;({
;()return res.json
,{
(onSettled: () => setUploading(false
```

```
;({
```

Setup clone const setupClone = useMutation({ mutationFn: async //
(data: any) => { const res = await fetch(‘/api/clone/setup’, { method:
‘POST’, headers: { ‘Content-Type’: ‘application/json’ }, body:
;({ { ;()JSON.stringify(data) }); return res.json

) return

```
<"div className="max-w-4xl mx-auto>
{/* Header */}
<"div className="text-center mb-8>
div className="inline-flex p-4 bg-gradient-to-br from-purple->
<"600 to-blue-600 rounded-full mb-4
</ "Brain className="h-12 w-12 text-white>
<div/>
h1 className="text-3xl font-bold text-white> إستنساخ الأم
<h1/>
"<p className="text-gray-400> قم بتدريب نسخة ذكية
<p/>
<div/>

<"div className="space-y-6>
{/* Voice Samples */}
<"Card className="bg-gray-800 border-gray-700>
<CardHeader>
<"CardTitle className="text-white flex items-center gap-2>
</ "Mic className="h-5 w-5 text-red-400>
عينات الصوت
<CardTitle/>
<CardHeader/>
<CardContent>
<"p className="text-gray-400 mb-4>
لتدريب النموذج 5
<p/>
div className="border-2 border-dashed border-gray-600>
<"rounded-lg p-8 text-center
Upload className="h-12 w-12 text-gray-500 mx-auto mb-4">
</
input>
"type="file
multiple
"*/accept="audio
"className="hidden
"id="voice-upload
} <= (onChange={(e
} (if (e.target.files
})uploadFiles.mutate
,(files: Array.from(e.target.files
'type: 'voice
;({
```

```
                                                {
                                                 {{
                                                   </
                                <"label htmlFor="voice-upload>
 <"Button variant="outline" className="cursor-pointer>
                                                        اختر ملفات الصوت
                                              <Button/>
                                                 <label/>
 p className="text-gray-500 text-sm mt-2">MP3, WAV, OGG>
                                                   <50MB)</p حتى(
                                                 <div/>
                                               <CardContent/>
                                                        <Card/>

                                                 {/* Photos */}
                    <"Card className="bg-gray-800 border-gray-700>
                                                 <CardHeader>
<"CardTitle className="text-white flex items-center gap-2>
                 </ "Image className="h-5 w-5 text-blue-400>
                                                           الصور
                                               <CardTitle/>
                                                <CardHeader/>
                                                <CardContent>
                            <"p className="text-gray-400 mb-4>
                                                          واجهة التوأم
                                                         <p/>
    div className="border-2 border-dashed border-gray-600>
                                          <"rounded-lg p-8 text-center
                                                    input>
                                             "type="file
                                                multiple
                                          "*/accept="image
                                        "className="hidden
                                         "id="photo-upload
                                        } <= (onChange={(e
                                    } (if (e.target.files
                                   })uploadFiles.mutate
                    ,(files: Array.from(e.target.files
                                       'type: 'photo
                                                  ;({
                                                      {
                                                       {{
                                                         </
                                <"label htmlFor="photo-upload>
 <"Button variant="outline" className="cursor-pointer>
                                                        اختر الصور
                                              <Button/>
                                                 <label/>
                                                      <div/>
                                               <CardContent/>
                                                        <Card/>

                                               {/* Documents */}
                    <"Card className="bg-gray-800 border-gray-700>
                                                 <CardHeader>
<"CardTitle className="text-white flex items-center gap-2>
                 </ "FileText className="h-5 w-5 text-green-400>
                                                          اعدة المعرفة
                                               <CardTitle/>
                                                <CardHeader/>
                                                <CardContent>
                            <"p className="text-gray-400 mb-4>
                                                          عرفها التوأم
```

```
<p/>
div className="border-2 border-dashed border-gray-600>
<"rounded-lg p-8 text-center
input>
"type="file
multiple
"accept=".pdf,.doc,.docx,.txt
"className="hidden
"id="doc-upload
} <= (onChange={(e
} (if (e.target.files
})uploadFiles.mutate
,(files: Array.from(e.target.files
'type: 'document
;({
{
{{
</
<"label htmlFor="doc-upload>
<"Button variant="outline" className="cursor-pointer>
اختر المستندات
<Button/>
<label/>
<div/>
<CardContent/>
<Card/>

{/* Personality Settings */}
<"Card className="bg-gray-800 border-gray-700>
<CardHeader>
<"CardTitle className="text-white flex items-center gap-2>
</ "Settings className="h-5 w-5 text-purple-400>
ادات الشخصية
<CardTitle/>
<CardHeader/>
<"CardContent className="space-y-4>
<div>
label className="text-gray-400 text-sm block mb->
<label/>رسالة الترحيب>"2
Textarea>
placeholder مساعد الرقمي لـ[اسمك]. كيف
"
"className="bg-gray-700 border-gray-600 text-white
</
<div/>
<div>
label className="text-gray-400 text-sm block mb-2>"وصف
<label/>
Textarea>
"...محترف، ودود، يحب مساعدة الآخرين"=placeholder
"className="bg-gray-700 border-gray-600 text-white
</
<div/>
<"div className="flex items-center justify-between>
<div>
<p/>الرد التلقائي<"p className="text-white>
p className="text-gray-500 text-sm> الساعة التوأم
<p/>
<div/>
</ Switch>
<div/>
<CardContent/>
<Card/>
```

```
                                        {/* Save Button */}
                                           Button>
className="w-full bg-gradient-to-r from-purple-600 to-blue-
                                                  "600 py-6 text-lg
                      {({})onClick={() => setupClone.mutate
                            {disabled={setupClone.isPending
                                                    <
                         </ "Brain className="h-5 w-5 ml-2>
                                                        م الرقمي
                                                 <Button/>
                                               <div/>
                                                 <div/>
                                                   <div/>

                                                      ); {

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


قيد المراجعة: SA 1020258841