Echelon MicroSaaS: Developer-Ready Specification

Executive Summary

This document outlines the MVP user flow, stack, and implementation steps for Echelon MicroSaaS. The goal is to enable clients to create, deploy, and manage custom AI agents via a self-serve portal. Stack: Base44 frontend, Supabase backend, Glyph chatbot, n8n Cloud for workflow automation, and LLM APIs for agent logic.

---

1. Complete User Flow (Screen-by-Screen)

MVP CORE PATH: Steps 1-8 below represent the essential user journey. All other features are Phase 2+.

Step 1: Client Login [MVP CORE]

Screen: Portal Login Page

- **URL**: portal.echelon.com/login
- **Elements**:
  - Echelon branding/logo
  - Email input field
  - Password input field
  - "Sign In" button
  - ~~"Forgot Password?" link~~ *(Phase 2+)*
- **Actions**:
  - User enters credentials
  - System validates against Supabase auth
  - Redirect to Dashboard on success
  - Show error message for invalid credentials
- **Data Collected**: email, password
- **Storage**: Supabase Auth table
- **Dev Notes**:
  - Use Supabase auth.signInWithPassword()
  - If Base44 has limited auth customization, implement error handling in JavaScript
  - Ensure redirect works properly after successful login

Step 2: Dashboard Overview [MVP CORE]

Screen: Portal Dashboard

- **URL**: portal.echelon.com/dashboard
- **Elements**:
  - Welcome message with user name
  - ~~Agent status cards (if agents exist)~~ *(Phase 2+)*
  - "Create New Agent" button (prominent, primary CTA)
  - ~~Recent activity feed~~ *(Phase 2+)*
  - Navigation menu (Dashboard, ~~Agents, Activity,~~ Help)
- **Actions**:
  - Click "Create New Agent" → Navigate to Onboarding
  - ~~View existing agent status~~ *(Phase 2+)*
  - ~~Access recent workflow logs~~ *(Phase 2+)*
- **Data Displayed**: User profile only for MVP
- **Storage**: Read from Supabase users table
- **Dev Notes**:
  - Keep dashboard minimal for MVP - focus on single "Create Agent" flow
  - If Base44 navigation is limited, use simple button-based navigation
  - Ensure prominent CTA button is visually distinct

Step 3: Agent Onboarding Start [MVP CORE]

Screen: Onboarding Introduction

- **URL**: portal.echelon.com/onboarding
- **Elements**:
  - "Let's Create Your AI Agent" header
  - Brief explanation of the process (2-3 sentences max)
  - Glyph chatbot interface (embedded)
  - ~~Progress indicator (Step 1 of 5)~~ *(Phase 2+)*
- **Actions**:
  - Chatbot initiates conversation automatically on page load
  - User responds to prompts in chat interface

- **Data Collected**: Initial user preferences
- **Storage**: Temporary session storage, then Supabase agents table
- **Dev Notes**:
  - If Glyph embedding is complex, start with iframe implementation
  - Ensure chat interface is mobile-responsive
  - Test auto-scroll behavior for chat messages
  - If Glyph has webhook limitations, consider polling for responses

Step 4: Goal Definition (Chatbot) [MVP CORE]

Screen: Chatbot Conversation Interface

- **Chatbot Flow**:

  ```
  Bot: "Hi! I'm here to help you create your AI agent. What's your name?"
  User: [Response] (Required field validation)

  Bot: "Nice to meet you, [Name]! What would you like your AI agent to help you with? For example:
  - Managing customer inquiries
  - Following up on leads
  - Creating reports
  - Something else?"
  User: [Response] (Free text, minimum 10 characters)

  Bot: "Great! Can you describe what success looks like when your agent is working well?"
  User: [Response] (Free text, minimum 10 characters)
  ```

- **Data Collected**:
  - agent_name (string, required)
  - primary_goal (text, required, min 10 chars)
  - success_criteria (text, required, min 10 chars)
- **Storage**: Supabase agents.goals (JSONB)
- **Dev Notes**:
  - Implement client-side validation before sending to Glyph
  - If Glyph doesn't support conditional logic, handle branching in frontend
  - Store partial responses in localStorage as backup (but clear on completion)
  - Add "Edit Previous Response" functionality if Glyph allows

Step 5: Tool Selection (Chatbot) [MVP CORE]

Screen: Same chatbot interface, new conversation phase

- **Chatbot Flow**:

  Bot: "Perfect! Now let's connect your agent to the tools you use. Which of these do you currently use?"

  [Display as buttons/checkboxes:]
  ☐ Slack
  ☐ Email (Gmail/Outlook)
  ☐ Google Calendar
  ☐ CRM (we'll ask which one next)
  ☐ None of these yet

  User: [Selections] (At least one required for MVP)

  Bot: "Great! I'll help you connect [Selected Tools]. This will require you to authorize access - you can revoke this anytime."

- **Actions**:
  - For each selected tool, show OAuth authorization link
  - Store connection tokens securely
  - Verify connection before proceeding
- **Data Collected**:
  - selected_tools (array, minimum 1 item)
  - tool_credentials (encrypted in separate table)
- **Storage**: Supabase integration_connections table
- **Dev Notes**:
  - **MVP LIMITATION**: Start with only Slack + Email for first version
  - If OAuth flow breaks Base44 interface, open in new tab with callback
  - Implement connection testing before moving to next step
  - If tool authorization fails, allow user to skip and continue
  - Store authorization status clearly for troubleshooting

Step 6: Behavior Configuration (Chatbot) [MVP CORE]

Screen: Continued chatbot conversation

- **Chatbot Flow**:

Bot: "Now let's set up how your agent should behave. How would you describe your communication style?"

[Display as buttons:]
- Professional and formal
- Friendly and casual
- Direct and concise

User: [Selection] (Required)

Bot: "Should your agent ask for approval before taking actions, or can it work automatically?"

[Display as buttons:]
- Always ask before acting
- Work automatically with daily summaries

User: [Selection] (Required)

- **Data Collected**:
  - communication_style (string, required)
  - approval_level (string, required)
  - ~~active_hours (object)~~ *(Phase 2+)*
- **Storage**: Supabase agents.preferences (JSONB)
- **Dev Notes**:
  - **MVP SIMPLIFICATION**: Removed active hours and complex approval levels
  - Use button selection instead of free text for consistency
  - Validate selection before proceeding to next step
  - If Glyph doesn't support buttons, use numbered options

Step 7: Review & Confirmation [MVP CORE]

Screen: Configuration Summary

- **Elements**:
  - Agent name and goal summary
  - Connected tools list (with connection status indicators)
  - Behavior preferences summary
  - ~~"Edit" buttons for each section~~ *(Phase 2+)*
  - "Deploy Agent" button (prominent)

    - "Start Over" button (secondary)
- **Actions**:
    - User reviews configuration
    - ~~Can edit any section (returns to chatbot)~~ *(Phase 2+)*
    - Click "Deploy Agent" to create
    - Click "Start Over" to restart onboarding
- **Data Collected**: Final confirmation timestamp
- **Storage**: Update Supabase agents.status to 'deploying'
- **Dev Notes**:
    - **MVP SIMPLIFICATION**: No editing capability - user must start over if changes needed
    - Clearly display all collected information for final review
    - Show tool connection status (connected/failed) with visual indicators
    - If Base44 has layout limitations, use simple list format
    - Ensure "Deploy" button is disabled until all validations pass

Step 8: Agent Deployment [MVP CORE]

Screen: Deployment Progress

- **Elements**:
    - Loading spinner
    - "Creating your agent..." message
    - Progress steps:
        - ✓ Configuration saved
        - ⏳ Connecting to tools
        - ⏳ Building workflows
        - ⏳ Agent ready
- **Actions**:
    - n8n Cloud workflow creation triggered via webhook
    - Real-time status updates (poll every 2 seconds)
    - Auto-redirect to Agent Dashboard on completion (or error page if failed)
- **Data Collected**: Deployment status and timestamps
- **Storage**: Supabase workflows table creation

- **Dev Notes**:
  - **CRITICAL**: Implement proper error handling for deployment failures
  - If real-time updates are complex, use simple polling every 3-5 seconds
  - Set maximum timeout of 2 minutes for deployment
  - If n8n API fails, store config and allow manual retry
  - Provide clear error messages and next steps if deployment fails

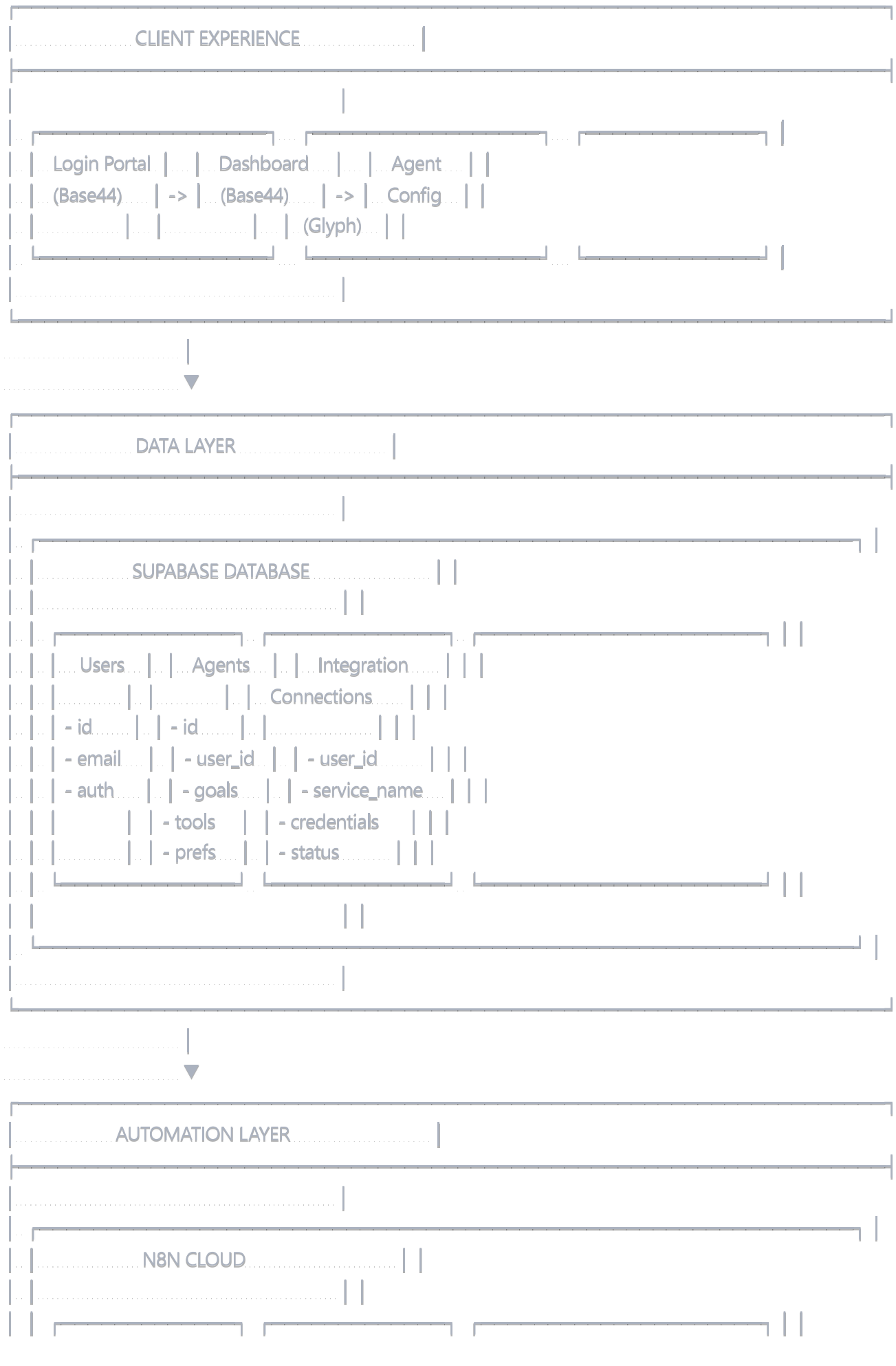
Step 9: Agent Dashboard [PHASE 2+]

Screen: Live Agent Interface

- **URL**: portal.echelon.com/agents/[agent_id]
- **Elements**:
  - Agent status indicator (Active/Paused/Error)
  - Basic "Agent is running" message
  - "Pause Agent" / "Resume Agent" buttons
  - "Create Another Agent" button
  - Simple activity log (last 5 actions)
- **Actions**:
  - Monitor basic agent status
  - Pause/resume agent
  - Navigate back to create new agent
- **Data Displayed**: Basic workflow status from Supabase
- **Storage**: Read from Supabase workflows, agents tables
- **Dev Notes**:
  - **PHASE 2+ FEATURE**: Keep this extremely simple for MVP
  - Focus on basic status display rather than detailed analytics
  - If Base44 has routing limitations, use query parameters instead of clean URLs
  - Implement basic error state handling

---

2. Stack Architecture Diagram


CLIENT EXPERIENCE
Login Portal
(Base44)
->
Dashboard
(Base44)
->
Agent
Config
(Glyph)
DATA LAYER
SUPABASE DATABASE
Users
- id
- email
- auth
Agents
- id
- user_id
- goals
- tools
- prefs
Integration
Connections
- user_id
- service_name
- credentials
- status
AUTOMATION LAYER
N8N CLOUD

Data Flow:

1. **User Input** → Base44 Portal → Glyph Chatbot
2. **Configuration Data** → Supabase Database
3. **Agent Trigger** → n8n Cloud Workflow Creation
4. **Workflow Execution** → External API Calls
5. **Results** → Supabase → Portal Dashboard

---

3. Chatbot Configuration Prompts

A. Welcome & Goal Setting [MVP CORE]

```
PROMPT_WELCOME = """
Hi there! I'm your AI agent setup assistant. I'll help you create a personalized AI agent that can automate your business processes.

This will take about 5 minutes, and I'll ask you about:
- What you want your agent to accomplish
- Which tools you'd like to connect
- How you want it to behave

Ready to get started? What's your name?
"""

PROMPT_GOAL_DISCOVERY = """
Great to meet you, {user_name}!

Let's start with the big picture. What's the main thing you'd like your AI agent to help you with?

Here are some examples to get you thinking:
- "Help me follow up with leads in my CRM"
- "Schedule meetings and send reminders"
- "Monitor customer support tickets and escalate urgent ones"
- "Create weekly reports from my project data"
- "Qualify incoming leads and route them to the right team"

What would be most valuable for you?
"""

PROMPT_SUCCESS_CRITERIA = """
Perfect! Now help me understand what success looks like.

When your agent is working well, what specific outcomes do you want to see?

For example:
- "No lead goes more than 2 hours without a response"
- "My team gets a summary every Friday of the week's activities"
- "Urgent issues are escalated to me within 15 minutes"
- "All new contacts are automatically added to my nurture sequence"

What's your version of success?
"""
```

B. Tool Selection & Setup [MVP CORE]

PROMPT_TOOL_SELECTION = """
Now let's connect your agent to the tools you use daily.

Which of these tools do you currently use? (Select all that apply)

💬 Communication:
- Slack
- Email (Gmail/Outlook)

📅 Scheduling:
- Google Calendar

📊 Customer Management:
- I have a CRM (we'll ask which one next)
- I don't use a CRM yet

Select the tools you'd like your agent to work with, or choose "None of these yet" if you want to start without integrations.
"""

PROMPT_TOOL_AUTHORIZATION = """
Excellent! I can see you use {selected_tools}.

For each tool, I'll need you to authorize the connection so your agent can access the necessary data.

Let's start with {first_tool}. I'll open an authorization window where you can securely connect your account.

Important: Your agent will only access the data needed for the tasks you've defined. You can revoke access at any time.

Ready to connect {first_tool}?
"""

C. Behavior Configuration [MVP CORE]

PROMPT_COMMUNICATION_STYLE = """
Now let's set up how your agent communicates.

When your agent sends messages or emails, what tone should it use?

Choose the style that best fits your brand:

🎯 Professional & Formal
"Thank you for your inquiry. I will process your request and provide an update within 24 hours."

😊 Friendly & Casual
"Hey! Thanks for reaching out. I'll take care of this and get back to you soon."

⚡ Direct & Concise
"Request received. Processing now. Update in 2 hours."

Which style feels right for your business?
"""

PROMPT_APPROVAL_LEVEL = """
How much independence should your agent have?

🛡️ Always ask before acting
Your agent will draft responses and actions but wait for your approval before executing them.

🤖 Work automatically with daily summaries
Your agent handles all defined tasks automatically and sends you a daily summary of what it accomplished.

What level of control feels right for you?
"""

D. Error Handling & Fallbacks [PHASE 2+]

PROMPT_ERROR_HANDLING = """
Let's set up what happens when your agent encounters issues.

Sometimes your agent might face situations it can't handle automatically - like an API being down, unclear customer requests, or unexpected errors.

How should your agent handle these situations?

🔄 Retry automatically
Try the action again after a brief delay, up to 3 times.

📧 Notify you immediately
Send you an email/Slack message when something goes wrong.

⏸️ Pause and wait for guidance
Stop the workflow and wait for your input before continuing.

📝 Log and continue
Record the error and continue with other tasks, then include it in your summary report.

What approach works best for you?
"""

PROMPT_FALLBACK_RULES = """
Finally, let's set up some fallback rules for edge cases.

When your agent encounters something unexpected, what should it do?

For unclear customer requests:
- Route to human support
- Ask clarifying questions
- Use best judgment with standard response

For missing information:
- Skip the task and flag it
- Use default values where appropriate
- Request the information and retry later

For system outages:
- Retry later automatically
- Switch to backup methods
- Notify you and pause

How would you like these handled?
"""

E. Review & Confirmation [MVP CORE]

PROMPT_REVIEW_CONFIG = """
Perfect! Let me summarize your agent configuration:

🎯 **Agent Goal**: {agent_goal}
✅ **Success Criteria**: {success_criteria}
🛠️ **Connected Tools**: {connected_tools}
💬 **Communication Style**: {communication_style}
⚙️ **Approval Level**: {approval_level}

Does this look correct?

You can:
- ✅ Deploy your agent now
- 🔄 Start over if you need to make changes

What would you like to do?
"""

PROMPT_DEPLOYMENT_START = """
Excellent! I'm now creating your AI agent.

Here's what's happening:
✅ Configuration saved
⏳ Connecting to your tools...
⏳ Building automation workflows...
⏳ Testing all connections...
⏳ Preparing your agent dashboard...

This usually takes 2-3 minutes. I'll let you know as soon as your agent is ready!
"""

---

4. Developer Handoff Document

Project Setup Requirements

Required Accounts & Access

○ **Base44**: Portal development platform ○ **Supabase**: Database and authentication (postgresql://...) ○ **n8n Cloud**: Workflow automation platform ○ **Glyph**: Chatbot conversation platform ○ **LLM API**: Claude or OpenAI API keys ○ **Integration APIs**: Slack, ConvertKit, Twilio, etc.

Environment Configuration

bash

```
# Environment Variables (.env)
SUPABASE_URL=your_supabase_url
SUPABASE_ANON_KEY=your_supabase_anon_key
N8N_API_KEY=your_n8n_api_key
GLYPH_API_KEY=your_glyph_api_key
CLAUDE_API_KEY=your_claude_api_key
OPENAI_API_KEY=your_openai_api_key
```

Database Schema (Supabase)

sql

```sql
-- Execute this SQL in Supabase SQL Editor

-- Enable UUID extension
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

-- Users table (managed by Supabase Auth)
CREATE TABLE users (
  id UUID REFERENCES auth.users(id) PRIMARY KEY,
  email VARCHAR UNIQUE NOT NULL,
  full_name VARCHAR,
  avatar_url VARCHAR,
  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW()
);

-- Agents table
CREATE TABLE agents (
  id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
  user_id UUID REFERENCES users(id) ON DELETE CASCADE,
  name VARCHAR NOT NULL,
  description TEXT,
  goals JSONB NOT NULL DEFAULT '{}',
  tools JSONB NOT NULL DEFAULT '[]',
  preferences JSONB NOT NULL DEFAULT '{}',
  status VARCHAR DEFAULT 'draft' CHECK (status IN ('draft', 'deploying', 'active', 'paused', 'error')),
  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW()
);

-- Workflows table
CREATE TABLE workflows (
  id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
  agent_id UUID REFERENCES agents(id) ON DELETE CASCADE,
  n8n_workflow_id VARCHAR UNIQUE,
  name VARCHAR NOT NULL,
  trigger_type VARCHAR NOT NULL,
  status VARCHAR DEFAULT 'active' CHECK (status IN ('active', 'paused', 'error')),
  last_run TIMESTAMP,
  run_count INTEGER DEFAULT 0,
  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW()
);
```

```sql
-- Integration connections table
CREATE TABLE integration_connections (
  id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
  user_id UUID REFERENCES users(id) ON DELETE CASCADE,
  service_name VARCHAR NOT NULL,
  connection_data JSONB NOT NULL,
  is_active BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW(),
  UNIQUE(user_id, service_name)
);

-- Workflow execution logs
CREATE TABLE workflow_logs (
  id UUID DEFAULT uuid_generate_v4() PRIMARY KEY,
  workflow_id UUID REFERENCES workflows(id) ON DELETE CASCADE,
  execution_id VARCHAR,
  status VARCHAR NOT NULL CHECK (status IN ('success', 'error', 'running')),
  input_data JSONB,
  output_data JSONB,
  error_message TEXT,
  execution_time INTEGER, -- milliseconds
  created_at TIMESTAMP DEFAULT NOW()
);

-- Row Level Security (RLS) Policies
ALTER TABLE users ENABLE ROW LEVEL SECURITY;
ALTER TABLE agents ENABLE ROW LEVEL SECURITY;
ALTER TABLE workflows ENABLE ROW LEVEL SECURITY;
ALTER TABLE integration_connections ENABLE ROW LEVEL SECURITY;
ALTER TABLE workflow_logs ENABLE ROW LEVEL SECURITY;

-- Policies
CREATE POLICY "Users can view own profile" ON users FOR SELECT USING (auth.uid() = id);
CREATE POLICY "Users can update own profile" ON users FOR UPDATE USING (auth.uid() = id);

CREATE POLICY "Users can view own agents" ON agents FOR SELECT USING (auth.uid() = user_id);
CREATE POLICY "Users can create own agents" ON agents FOR INSERT WITH CHECK (auth.uid() = user_id);
CREATE POLICY "Users can update own agents" ON agents FOR UPDATE USING (auth.uid() = user_id);
CREATE POLICY "Users can delete own agents" ON agents FOR DELETE USING (auth.uid() = user_id);

-- Similar policies for other tables...
```

API Endpoints Specification

Base44 Portal Routes

```
/login - Authentication page
/dashboard - Main dashboard
/onboarding - Agent creation wizard
/agents - Agent management
/agents/:id - Individual agent dashboard
/integrations - Connected tools management
/activity - Workflow execution logs
/help - Documentation and support
```

Supabase API Calls

```javascript
// Authentication
supabase.auth.signInWithPassword({ email, password })
supabase.auth.signUp({ email, password })

// Agents CRUD
supabase.from('agents').select('*').eq('user_id', user_id)
supabase.from('agents').insert({ user_id, name, goals, tools, preferences })
supabase.from('agents').update({ preferences }).eq('id', agent_id)

// Workflows
supabase.from('workflows').select('*').eq('agent_id', agent_id)
supabase.from('workflow_logs').select('*').eq('workflow_id', workflow_id)
```

n8n Cloud Integration

```javascript
```

```
// Create workflow
POST /api/v1/workflows
{
  "name": "Agent Workflow - {agent_name}",
  "nodes": [...], // Generated based on agent config
  "connections": {...}
}

// Trigger workflow
POST /api/v1/workflows/{workflow_id}/activate
POST /api/v1/executions

// Monitor workflows
GET /api/v1/workflows/{workflow_id}/executions
```

Development Priorities

Phase 1: Core Infrastructure (Week 1-2) [MVP CORE]

1. **Database Setup**
   - Create Supabase project
   - Run database schema
   - Set up RLS policies
   - Test authentication
2. **Base44 Portal**
   - Create login page
   - Set up dashboard template
   - Configure routing
   - Implement authentication flow
3. **Basic Integration**
   - Connect Base44 to Supabase
   - Test user registration/login
   - Create agent CRUD operations

Phase 2: Chatbot Integration (Week 3-4) [MVP CORE]

1. **Glyph Setup**
   - Configure conversation flows

- Implement prompt templates
- Set up webhook endpoints
- Test conversation logic

2. **Data Collection**
   - Build forms for agent configuration
   - Implement data validation
   - Store configuration in Supabase
   - Create review/edit functionality

Phase 3: Workflow Automation (Week 5-8) [MVP CORE]

1. **n8n Cloud Setup**
   - Create workflow templates
   - Implement dynamic workflow generation
   - Set up external API connections
   - Test workflow execution

2. **n8n Cloud Setup** [MVP CORE]
   - Create basic workflow templates for Slack + Email integrations only
   - Implement simple workflow generation (no complex dynamic logic)
   - Set up external API connections for MVP tools
   - Test workflow execution with hardcoded examples

Phase 4: Testing & Polish (Week 9-11) [MVP CORE]

1. **End-to-End Testing**
   - Complete user journey testing
   - Error scenario testing
   - Performance optimization
   - Security review

2. **Documentation**
   - User guides
   - API documentation
   - Deployment procedures
   - Troubleshooting guides

Success Metrics [MVP CORE]

○ User can complete full onboarding in <5 minutes
○ Agent deployment succeeds 95% of the time for Slack + Email integrations
○ Basic workflow execution works end-to-end
○ Zero data loss during configuration
○ Mobile-responsive on all core screens

Post-MVP Features (Phase 2+)

○ Multi-agent management
○ Advanced integration tools (CRM, project management)
○ Detailed analytics and reporting
○ Configuration editing capabilities
○ Advanced error handling and retry logic
○ Custom integration development
○ Team collaboration features

Next Actions for Developers

Immediate (This Week) [MVP CORE]

1. **Set up development environment**
   - Clone repository (or create new Base44 project)
   - Install dependencies
   - Configure environment variables
   - Verify API access for Supabase, n8n Cloud, Glyph
2. **Create database schema**
   - Run Supabase SQL scripts (provided below)
   - Test authentication with dummy user
   - Verify table relationships
   - Set up sample data for testing
3. **Build login flow**
   - Create login page in Base44
   - Implement Supabase auth
   - Add error handling for invalid credentials
   - Test user registration (if self-registration needed)

Week 2 [MVP CORE]

1. **Dashboard implementation**
   - Create minimal dashboard layout
   - Add "Create New Agent" button (primary CTA)
   - Display welcome message with user name
   - Implement basic navigation
2. **Glyph integration setup**
   - Set up Glyph account and conversation flows
   - Configure webhook endpoints for data collection
   - Test basic chatbot embedding in Base44
   - Implement session management for conversation state

Week 3 [MVP CORE]

1. **Onboarding flow**
   - Build onboarding page with embedded Glyph
   - Implement data collection from chatbot responses
   - Store configuration data in Supabase
   - Add validation for required fields
2. **Tool authorization (MVP: Slack + Email only)**
   - Set up OAuth flows for Slack
   - Implement email connection (Gmail/Outlook)
   - Store credentials securely in Supabase
   - Test connection verification

Week 4-6 [MVP CORE]

1. **n8n Cloud workflow setup**
   - Create basic workflow templates
   - Implement workflow generation from user config
   - Set up LLM API integration (Claude or OpenAI)
   - Test end-to-end workflow execution
2. **Deployment flow**
   - Build deployment progress page

- Implement workflow creation trigger
- Add error handling for deployment failures
- Create basic agent status dashboard

Technical Considerations [CRITICAL FOR MVP]

Platform Limitations & Workarounds

Base44 Limitations:

- If custom routing is limited, use query parameters: ?page=onboarding&step=3
- If JavaScript execution is restricted, implement logic server-side with Supabase functions
- If OAuth redirects break interface, open in new tab with postMessage communication
- If real-time updates aren't supported, use polling every 3-5 seconds

Glyph Integration Issues:

- If webhook delays are >2 seconds, implement loading states
- If conditional logic is limited, handle branching in Base44 frontend
- If embedding is complex, start with iframe implementation
- If conversation state is lost, store backup in localStorage

n8n Cloud Constraints:

- Start with simple, linear workflows only
- If dynamic workflow generation is complex, use predefined templates
- If API rate limits are hit, implement queuing system
- If workflow fails, store config for manual retry

Error Handling Strategy [MVP FOCUS]

1. **User-Facing Errors:**
   - Clear, actionable error messages
   - "Try Again" buttons for recoverable errors
   - "Contact Support" for system failures
   - Progress preservation during errors
2. **System Errors:**
   - Log all errors to Supabase for debugging

- Implement graceful degradation
- Provide manual workarounds
- Alert development team for critical failures

3. **Data Integrity:**
   - Validate all user input client-side and server-side
   - Implement transaction rollbacks for failed deployments
   - Regular backup of configuration data
   - Audit trail for all user actions

Security Requirements [NON-NEGOTIABLE]

○ All API keys stored as environment variables (never in code)
○ OAuth tokens encrypted before database storage
○ HTTPS enforced on all endpoints
○ Supabase RLS policies properly configured
○ Input sanitization for all user data
○ Rate limiting on API endpoints
○ Regular security audit of dependencies

Deployment Checklist [MVP LAUNCH]

Pre-Launch Requirements

○ All MVP user flows tested end-to-end
○ Database schema deployed to production
○ Environment variables configured
○ OAuth applications registered for production domains
○ Error logging and monitoring set up
○ Basic performance testing completed
○ Mobile responsiveness verified
○ User acceptance testing with 2-3 beta users

Launch Day

○ Production database backup
○ Monitoring dashboards active
○ Support documentation ready
○ Rollback plan prepared
○ Team communication plan active

Post-Launch (Week 1)

○ Monitor error rates and user completion
○ Collect user feedback on onboarding experience
○ Track agent deployment success rates
○ Identify most common support requests
○ Plan Phase 2+ features based on usage data

Migration Path to Emergence AI [FUTURE PLANNING]

- Document all n8n workflows with detailed specifications
- Maintain clean separation between data and logic layers
- Use standardized prompt templates that can be easily ported
- Keep configuration schema flexible for future enhancements
- Build modular components for easy platform migration
- Regular backup of all workflow definitions and user data

Final Implementation Notes

MVP Success Criteria

The MVP is considered successful when a user can:

1. **Log in** to the portal without issues
2. **Complete onboarding** via chatbot in under 5 minutes
3. **Connect at least one tool** (Slack or Email) successfully
4. **Deploy an agent** that creates a working n8n workflow
5. **See basic status** of their deployed agent

Critical Path Items

○ **Week 1**: Database + Authentication working ○ **Week 2**: Chatbot integration functional ○ **Week 3**: Tool connections working for Slack/Email ○ **Week 4**: n8n workflow creation automated ○ **Week 5-6**: End-to-end testing and polish

Risk Mitigation

High-Risk Items:

- n8n Cloud API integration complexity

- OAuth flow integration with Base44
- Glyph chatbot embedding and data collection
- Real-time status updates during deployment

Mitigation Strategies:

- Start with simplest possible implementations
- Build fallback options for each integration
- Test each component in isolation before integration
- Maintain clear rollback procedures

Support Documentation Required

○ **User Guide**: How to use the portal and create agents ○ **Developer Setup**: Environment configuration and deployment ○ **API Documentation**: All endpoint specifications and examples ○ **Troubleshooting**: Common issues and solutions ○ **Migration Guide**: Preparation for Emergence AI transition

This specification provides a complete roadmap for MVP development with clear priorities, technical constraints, and success criteria. The focus on simplicity and core functionality ensures rapid time-to-market while maintaining quality standards.