🚀 COMPLETE DEVELOPER HANDOVER DOCUMENT

QuantumLeap AI Website - Full Technical Documentation

Document Version: 2.0
Date Created: November 11, 2025
Last Updated: November 11, 2025
Project: QuantumLeap AI Website
Repository: https://github.com/AICyberSecurity911/WBsite.git
Branch: semi-final
Live URL: https://quantumleapai.abacusai.app

📋 TABLE OF CONTENTS

📝 EXECUTIVE SUMMARY

What is QuantumLeap AI?

QuantumLeap AI is a B2B SaaS website offering AI-powered business services:
- **AI Workforce Solutions** - AI employee replacements
- **Intelligent Automation** - Process automation services
- **Cyber Intelligence** - Advanced threat detection
- **Beyond Background Checks** - Deep background investigations
- **Business Transformation** - Full business digitalization

Current Status

✅ **Production Ready** - Fully functional and deployed
✅ **NPM-Only** - Complete migration from Yarn to npm
✅ **Quantum Gradient Dark Theme** - Modern dark-mode-first design
✅ **23 Pages** - Complete website with admin dashboard
✅ **6 API Integrations** - Gmail, Calendar, TidyCal, HIBP, Discord, Firebase
✅ **4 Interactive Calculators** - ROI calculators with lead capture
✅ **Admin Dashboard** - Full CRM and content management

Tech Stack Overview

- **Framework:** Next.js 14.2.28 (App Router)
- **Language:** TypeScript 5.2.2
- **Styling:** Tailwind CSS 3.3.3 + Custom Design System
- **UI Library:** shadcn/ui (50+ components)
- **Database:** PostgreSQL (Supabase) + Prisma ORM 6.7.0
- **Animations:** Framer Motion 10.18.0
- **State:** Zustand 5.0.3
- **Package Manager:** npm (100% Yarn-free)

Key Metrics

- **270+ Files** tracked in Git
- **1.3GB** project size (excluding dependencies)
- **211 Dependencies** + 18 dev dependencies
- **50+ UI Components** (shadcn/ui)
- **23 Pages** (18 public + 5 admin)
- **20+ API Routes**
- **6 Database Models**
- **4 Interactive Calculators**
- **Build Time:** ~45 seconds
- **Lighthouse Score Target:** 90+ across all metrics

🚀 QUICK START GUIDE

Prerequisites

```
# Required software
- Node.js 18.x or higher
- npm 9.x or higher
- PostgreSQL 14+ (or Supabase account)
- Git

# Optional but recommended
- VS Code with extensions:
  - ESLint
  - Prettier
  - Tailwind CSS IntelliSense
  - Prisma
```

Initial Setup (New Developer)

```
# 1. Clone the repository
git clone https://github.com/AICyberSecurity911/WBsite.git
cd WBsite

# 2. Checkout the correct branch
git checkout semi-final

# 3. Navigate to Next.js workspace
cd nextjs_space

# 4. Install dependencies (IMPORTANT: Use --legacy-peer-deps)
npm install --legacy-peer-deps

# 5. Set up environment variables
# Copy credentials from /home/ubuntu/quantumleap_credentials.json
# Create .env.local file (see Environment Configuration section)
cp .env.example .env.local
# Then edit .env.local with actual values

# 6. Generate Prisma client
npx prisma generate

# 7. Run database migrations (if needed)
npx prisma db push

# 8. Start development server
npm run dev

# 9. Open browser
# Navigate to http://localhost:3000
```

Daily Development Workflow

```
# 1. Pull latest changes
git pull origin semi-final

# 2. Install any new dependencies
npm install --legacy-peer-deps

# 3. Start dev server
npm run dev

# 4. Make changes and test locally

# 5. Build to verify no errors
npm run build

# 6. Commit and push
git add .
git commit -m "Description of changes"
git push origin semi-final
```

Quick Commands Reference

```
# Development
npm run dev              # Start dev server (localhost:3000)
npm run build            # Production build
npm run start            # Start production server
npm run lint             # Run ESLint

# Database
npx prisma studio        # Open Prisma Studio (database GUI)
npx prisma generate      # Generate Prisma client
npx prisma db push       # Push schema changes to database
npx prisma migrate dev   # Create and apply migration

# Deployment
npm run build            # Build for production
# Then deploy to Vercel/Netlify/Abacus.AI
```

📊 PROJECT OVERVIEW

Business Model

Target Audience: SMBs and Enterprises seeking AI transformation

Primary Services:
1. **AI Workforce** - Replace employees with AI agents
2. **Intelligent Automation** - Automate business processes
3. **Cyber Intelligence** - Advanced threat detection and monitoring
4. **Beyond Background Checks** - Deep investigations and vetting
5. **Business Transformation** - Complete digital transformation consulting

Revenue Model: Service-based (consultation bookings → custom quotes)

User Journeys

Journey 1: SMB Owner (Homepage → SMB Landing → Consultation)

```
1. Land on homepage (/)
2. Click "View Small Business Solutions"
3. View SMB landing page (/smb)
4. Interact with calculator
5. Submit lead form
6. Receive email with next steps
7. Book consultation (/consultation)
```

Journey 2: Enterprise Decision Maker (Service Page → Calculator → Contact)

```
1. Search for "AI workforce solutions"
2. Land on /ai-workforce
3. Read service details
4. Use AI Team Calculator
5. See ROI projections
6. Submit contact form
7. Receive Discord notification (internal)
8. Sales team follows up
```

Journey 3: Security-Conscious Buyer (Breach Check → Background Checks → Consultation)

```
1. Land on homepage
2. Use email breach checker
3. Discover compromised accounts
4. Navigate to /background-checks
5. Learn about deep investigations
6. Book consultation
```

Conversion Funnels

Primary Funnel:

```
Homepage → Service Page → Calculator → Lead Capture → Consultation → Sale
```

Secondary Funnel:

```
Homepage → Exit Intent Popup → Lead Magnet → Email Nurture → Consultation
```

Blog Funnel:

```
Blog Post → Internal Links → Service Page → Lead Capture
```

Key Performance Indicators (KPIs)

- **Lead Capture Rate:** Target 5-10% of visitors
- **Calculator Completion Rate:** Target 30-40%
- **Consultation Booking Rate:** Target 15-25% of leads
- **Email Open Rate:** Target 30-40%

- **Page Load Time:** Target < 3 seconds
- **Bounce Rate:** Target < 50%

🏗️ TECHNICAL ARCHITECTURE

High-Level Architecture

```
┌─────────────────────────────────────────────────────────────┐
│                        USER BROWSER                          │
└─────────────────────────────────────────────────────────────┘
                     │
                     ▼
┌─────────────────────────────────────────────────────────────┐
│                 NEXT.JS 14 (APP ROUTER)                      │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐      │
│  │   Pages      │  │     API      │  │  Components  │      │
│  │ (18 pub +    │  │  Routes      │  │   (50+ UI)   │      │
│  │  5 admin)    │  │  (20+ APIs)  │  │              │      │
│  └──────────────┘  └──────────────┘  └──────────────┘      │
└─────────────────────────────────────────────────────────────┘
                     │
                     ▼
┌─────────────────────────────────────────────────────────────┐
│                   PRISMA ORM LAYER                           │
└─────────────────────────────────────────────────────────────┘
                     │
                     ▼
┌─────────────────────────────────────────────────────────────┐
│               POSTGRESQL (SUPABASE)                          │
│  ┌─────────────────────────────────────────────────┐        │
│  │ Tables: leads, calculator_responses,            │        │
│  │ contact_submissions, admin_users,               │        │
│  │ blog_api_keys, consultation_bookings            │        │
│  └─────────────────────────────────────────────────┘        │
└─────────────────────────────────────────────────────────────┘

┌─────────────────────────────────────────────────────────────┐
│                THIRD-PARTY INTEGRATIONS                      │
├────────────────┬────────────────┬────────────────┬─────────┤
│  Gmail API     │ Google Cal     │    TidyCal     │  HIBP   │
│  (Email)       │  (Booking)     │   (Booking)    │(Breach) │
├────────────────┼────────────────┼────────────────┼─────────┤
│  Discord       │   Firebase     │  Google Drive  │         │
│  (Notify)      │   (Storage)    │   (Storage)    │         │
└────────────────┴────────────────┴────────────────┴─────────┘
```

Data Flow

Lead Capture Flow

```
1. User fills calculator form
2. POST /api/calculator/recommendations
3. Calculate ROI projections
4. Store in calculator_responses table
5. Create/update lead record
6. Send confirmation email (Gmail API)
7. Send Discord notification (Webhook)
8. Return recommendations to frontend
9. Show success modal
```

Consultation Booking Flow

```
1. User submits consultation form
2. POST /api/consultation
3. Create consultation_bookings record
4. Create Google Calendar event
5. Create TidyCal booking
6. Send confirmation email
7. Send Discord notification
8. Redirect to /confirmation
```

Admin Authentication Flow

```
1. Admin enters credentials
2. POST /api/admin/auth/login
3. Verify against admin_users table
4. Generate JWT token
5. Set HTTP-only cookie
6. Return success + user data
7. Redirect to /admin dashboard
```

Technology Decisions

Why Next.js 14 App Router?

- **Server Components:** Better performance, smaller bundle size
- **Built-in API Routes:** No need for separate backend
- **File-based Routing:** Intuitive project structure
- **Image Optimization:** Automatic next/image optimization
- **SEO-Friendly:** Server-side rendering for better SEO

Why Prisma ORM?

- **Type Safety:** Full TypeScript support
- **Migrations:** Easy database versioning
- **Studio GUI:** Visual database browser
- **Query Performance:** Optimized SQL generation
- **Developer Experience:** Great documentation

Why Supabase (PostgreSQL)?

- **Managed Service:** No server management
- **Real-time:** Built-in real-time subscriptions

- **Authentication:** Built-in auth (not used yet)
- **Storage:** Built-in file storage (not used yet)
- **Pricing:** Free tier for development

Why shadcn/ui?

- **Not a Package:** Copy-paste components (no dependency)
- **Customizable:** Full control over components
- **Accessible:** Built on Radix UI primitives
- **Tailwind-based:** Consistent with our styling approach
- **Type-safe:** Full TypeScript support

Why Framer Motion?

- **Smooth Animations:** Physics-based animations
- **Declarative API:** Easy to use and maintain
- **Performance:** GPU-accelerated animations
- **Gestures:** Built-in gesture support
- **SSR Compatible:** Works with Next.js SSR

🎨 DESIGN SYSTEM (QUANTUM GRADIENT DARK)

Color Palette

Core Colors (Always Use These)

```
/* Background Colors */
--qgd-bg: #07070b          /* Main background (deep space black) */
--qgd-card: #0c0c12        /* Card backgrounds */
--qgd-muted: #1a1a24       /* Muted elements */

/* Foreground Colors */
--qgd-fg: #f6f7ff          /* Main text (off-white) */
--qgd-fg-muted: #9ca3af    /* Secondary text (gray) */

/* Primary Colors */
--qgd-primary: #5312c4     /* Primary actions (deep purple) */
--qgd-ring: #7c3aed        /* Focus rings (bright purple) */

/* Accent Colors */
--qgd-accent: #ff7f50      /* Copper/coral accent */
--qgd-accent-hover: #ff6347 /* Copper hover state */

/* Border Colors */
--qgd-border: #2c2c3d      /* Card borders */
--qgd-border-hover: #3d3d52 /* Border hover state */
```

Usage Guidelines

Backgrounds:

```
// Page backgrounds
className="bg-qgd-bg"

// Card backgrounds
className="bg-qgd-card"

// Muted sections (sidebar, secondary areas)
className="bg-qgd-muted"
```

Text:

```
// Primary text
className="text-qgd-fg"

// Secondary text (descriptions, labels)
className="text-qgd-fg-muted"

// Accent text (CTAs, links)
className="text-qgd-accent"
```

Buttons:

```
// Primary CTA buttons
className="bg-qgd-primary hover:bg-qgd-ring"

// Secondary buttons
className="bg-qgd-ring/20 hover:bg-qgd-ring/30"

// Accent buttons
className="bg-qgd-accent hover:bg-qgd-accent-hover"
```

Cards:

```
// Standard card
className="bg-qgd-card border border-qgd-border"

// Hover state
className="hover:border-qgd-border-hover hover:shadow-qgd"
```

Animated Copper Flame Borders

All cards should have animated copper flame borders using the `FlameBorder` component:

```
import { FlameBorder } from '@/components/ui/flame-border'

<FlameBorder className="relative">
  {/* Card content */}
</FlameBorder>
```

Parameters:
- **Thickness:** 3px
- **Length:** 150 segments
- **Speed:** 28s cycle

- **Color:** `#b87333` (copper)
- **Glow:** `rgba(184, 115, 51, 0.6)` with `mix-blend-screen`

Card Glow Effects

All cards have permanent glow with purple hover intensification:

```
/* Base glow (always visible) */
box-shadow: 0 0 20px rgba(124, 58, 237, 0.3);

/* Hover glow (intensified) */
&:hover {
  box-shadow: 0 0 30px rgba(124, 58, 237, 0.6);
}
```

Implementation:

```
<div className="shadow-[0_0_20px_rgba(124,58,237,0.3)] hover:shadow-
[0_0_30px_rgba(124,58,237,0.6)] transition-shadow duration-300">
  {/* Card content */}
</div>
```

Typography

Font System

```
// Primary font: Inter (body text, UI)
font-family: 'Inter', sans-serif

// Secondary font: Manrope (headings)
font-family: 'Manrope', sans-serif
```

Font Scales

```
// Headings
<h1 className="text-5xl font-bold">     {/* 48px */}
<h2 className="text-4xl font-bold">     {/* 36px */}
<h3 className="text-3xl font-semibold"> {/* 30px */}
<h4 className="text-2xl font-semibold"> {/* 24px */}
<h5 className="text-xl font-medium">    {/* 20px */}

// Body Text
<p className="text-base">               {/* 16px - default */}
<p className="text-sm">                 {/* 14px - secondary */}
<p className="text-xs">                 {/* 12px - labels */}

// Display Text
<h1 className="text-6xl font-black">    {/* 60px - hero headings */}
<h1 className="text-7xl font-black">    {/* 72px - landing page */}
```

Line Heights

```
// Tight (headings)
className="leading-tight"  // 1.25

// Normal (body)
className="leading-normal" // 1.5

// Relaxed (long-form content)
className="leading-relaxed" // 1.75
```

Spacing System

```
// Use Tailwind's spacing scale (4px base)
space-y-2  // 8px
space-y-4  // 16px
space-y-6  // 24px
space-y-8  // 32px
space-y-12 // 48px
space-y-16 // 64px
space-y-20 // 80px
space-y-24 // 96px
```

Standard Section Spacing:

```
// Between sections on a page
<section className="py-16 md:py-20 lg:py-24">

// Container padding
<div className="px-4 sm:px-6 lg:px-8">

// Card padding
<div className="p-6 lg:p-8">
```

Animation Guidelines

Standard Transitions

```
// Hover effects
transition-all duration-300

// Focus states
transition-colors duration-200

// Scale effects
transition-transform duration-300 hover:scale-105

// Opacity changes
transition-opacity duration-500
```

Framer Motion Patterns

```
// Fade in on scroll
<motion.div
  initial={{ opacity: 0, y: 20 }}
  whileInView={{ opacity: 1, y: 0 }}
  viewport={{ once: true }}
  transition={{ duration: 0.6 }}
>

// Staggered children
<motion.div
  variants={{
    hidden: { opacity: 0 },
    show: {
      opacity: 1,
      transition: {
        staggerChildren: 0.1
      }
    }
  }}
  initial="hidden"
  whileInView="show"
>

// Scale on hover
<motion.div
  whileHover={{ scale: 1.05 }}
  transition={{ type: "spring", stiffness: 300 }}
>
```

Responsive Design

Breakpoints

```
// Tailwind breakpoints
sm:  640px  // Small devices (phones landscape)
md:  768px  // Medium devices (tablets)
lg:  1024px // Large devices (desktops)
xl:  1280px // Extra large devices (large desktops)
2xl: 1536px // 2X large devices (ultra-wide)
```

Mobile-First Approach

Always design for mobile first, then enhance for larger screens:

```
// ❌ Wrong (desktop-first)
<div className="lg:text-base md:text-sm text-xs">

// ✅ Correct (mobile-first)
<div className="text-xs md:text-sm lg:text-base">
```

Common Responsive Patterns

```
// Grid layouts
<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">

// Flex direction
<div className="flex flex-col lg:flex-row gap-4">

// Text alignment
<h1 className="text-center lg:text-left">

// Hide/show elements
<div className="hidden lg:block">  // Desktop only
<div className="block lg:hidden">  // Mobile only
```

Design Tokens Reference

```
// borders-qgd.css (in globals.css)
.border-qgd {
  @apply border-qgd-border;
}

.border-qgd-hover {
  @apply border-qgd-border-hover;
}

// Shadows
.shadow-qgd {
  box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.5), 0 2px 4px -1px rgba(0, 0, 0, 0.3);
}

.shadow-qgd-glow {
  box-shadow: 0 0 20px rgba(124, 58, 237, 0.3);
}

// Gradients
.gradient-qgd-primary {
  background: linear-gradient(135deg, #5312c4 0%, #7c3aed 100%);
}

.gradient-qgd-accent {
  background: linear-gradient(135deg, #ff7f50 0%, #ff6347 100%);
}
```

📁 CODEBASE STRUCTURE

Directory Overview

```
quantumleap_io/
├── nextjs_space/                  # Main Next.js application
│   ├── app/                       # Next.js 14 App Router
│   │   ├── (pages)/               # Page routes
│   │   ├── api/                   # API routes
│   │   ├── layout.tsx             # Root layout
│   │   ├── page.tsx               # Homepage
│   │   └── globals.css            # Global styles
│   ├── components/                # React components
│   │   ├── calculator/            # Calculator components (4)
│   │   ├── layout/                # Header, Footer
│   │   ├── sections/              # Page sections
│   │   ├── smb/                   # SMB-specific components
│   │   └── ui/                    # shadcn/ui components (50+)
│   ├── lib/                       # Utilities & integrations
│   ├── hooks/                     # Custom React hooks
│   ├── types/                     # TypeScript types
│   ├── public/                    # Static assets
│   ├── prisma/                    # Database schema
│   ├── scripts/                   # Setup scripts
│   ├── database/                  # SQL files
│   ├── package.json               # Dependencies
│   ├── tsconfig.json              # TypeScript config
│   ├── tailwind.config.ts         # Tailwind config
│   └── next.config.js             # Next.js config
├── Documentation/                 # All .md and .pdf files
└── Uploads/                       # User-uploaded assets
```

App Directory (`app/`)

```
app/
├── about-us/
│   └── page.tsx                 # About Us page
├── admin/                       # Admin dashboard
│   ├── blog-api/
│   ├── calendar/
│   ├── crm/
│   ├── leads/
│   ├── login/
│   ├── tidycal/
│   ├── layout.tsx               # Admin layout with auth
│   └── page.tsx                 # Admin dashboard
├── ai-workforce/
│   └── page.tsx                 # AI Workforce service
├── api/                         # API routes (20+)
│   ├── admin/                   # Admin APIs
│   ├── automation-calc/
│   ├── automation-lead/
│   ├── blog/
│   ├── breach-check/
│   ├── calculator/
│   ├── consultation/
│   ├── contact-support/
│   ├── cron-send-pending/
│   ├── lead-magnet/
│   ├── leads/
│   └── tidycal-webhook/
├── background-checks/
│   └── page.tsx                 # Background Checks service
├── blog/
│   ├── [slug]/
│   │   └── page.tsx             # Dynamic blog post
│   └── page.tsx                 # Blog listing
├── business-transformation/
│   └── page.tsx                 # Business Transformation
├── confirmation/
│   └── page.tsx                 # Booking confirmation
├── consultation/
│   └── page.tsx                 # Consultation booking
├── coral-reef-demo/
│   └── page.tsx                 # Design system demo
├── cyber-intelligence/
│   └── page.tsx                 # Cyber Intelligence service
├── intelligent-automation/
│   ├── contact/
│   │   └── page.tsx             # Automation contact form
│   └── page.tsx                 # Automation service
├── privacy/
│   └── page.tsx                 # Privacy policy
├── smb/
│   ├── metadata.ts              # SMB page metadata
│   └── page.tsx                 # SMB landing page
├── terms/
│   └── page.tsx                 # Terms of service
├── layout.tsx                   # Root layout
├── page.tsx                     # Homepage
└── globals.css                  # Global CSS with design tokens
```

Components Directory (`components/`)

```
components/
├── calculator/                   # Calculator components
│   ├── ai-team-calculator.tsx   # AI Team ROI calculator
│   ├── automation-calculator.tsx # Automation ROI calculator
│   ├── hiring-risk-calculator.tsx # Hiring Risk calculator
│   └── profit-potential-calculator.tsx # Profit calculator
├── layout/                       # Layout components
│   ├── header.tsx                # Main navigation
│   └── footer.tsx                # Footer with newsletter
├── sections/                     # Page sections
│   ├── advisory-board-section.tsx
│   ├── ai-employees-section.tsx
│   ├── blog-section.tsx
│   ├── calculator-section.tsx
│   ├── coral-reef-features.tsx
│   ├── coral-reef-hero.tsx
│   ├── faq-section.tsx
│   ├── founder-story-section.tsx
│   ├── guarantee-section.tsx
│   ├── hero-section.tsx
│   ├── home-testimonials-carousel.tsx
│   ├── problem-section.tsx
│   ├── split-hero-section.tsx
│   ├── testimonials-section.tsx
│   ├── tldr-section.tsx
│   ├── trust-bar-section.tsx
│   └── value-proposition-section.tsx
├── smb/                          # SMB-specific components
│   ├── animated-background.tsx   # Canvas particle system
│   ├── animated-button.tsx       # Animated CTA buttons
│   ├── hero-section.tsx          # SMB hero section
│   ├── scroll-progress.tsx       # Scroll progress bar
│   ├── theme-context.tsx         # Theme provider
│   └── theme-toggle.tsx          # Dark/light toggle
├── ui/                           # shadcn/ui components (50+)
│   ├── accordion.tsx
│   ├── alert-dialog.tsx
│   ├── avatar.tsx
│   ├── badge.tsx
│   ├── button.tsx
│   ├── card.tsx
│   ├── checkbox.tsx
│   ├── dialog.tsx
│   ├── dropdown-menu.tsx
│   ├── flame-border.tsx          # Animated flame border
│   ├── form.tsx
│   ├── input.tsx
│   ├── label.tsx
│   ├── select.tsx
│   ├── table.tsx
│   ├── tabs.tsx
│   ├── toast.tsx
│   ├── tooltip.tsx
│   └── [40+ more components]
├── automation-exit-intent.tsx    # Exit popup for automation
├── background-checks-exit-intent.tsx
├── business-transformation-exit-intent.tsx
├── exit-intent-provider.tsx      # Exit intent manager
├── lead-magnet-modal.tsx         # Lead magnet popup
└── theme-provider.tsx            # Theme context provider
```

Lib Directory (`lib/`)

```
lib/
├── admin-auth.ts           # Admin JWT authentication
├── calculator.ts           # Calculator logic
├── db.ts                   # Prisma client singleton
├── discord-automation.ts   # Discord webhook automation
├── discord.ts              # Discord utilities
├── email-automation.ts     # Email automation logic
├── email.ts                # Email sending utilities
├── firebase-admin.ts       # Firebase Admin SDK
├── google-calendar.ts      # Google Calendar integration
├── google-drive.ts         # Google Drive integration
├── schema-helpers.ts       # Zod validation schemas
├── supabase.ts             # Supabase client
├── tidycal.ts              # TidyCal integration
├── types.ts                # TypeScript types
└── utils.ts                # Utility functions
```

Key Files Explained

`app/layout.tsx` - Root Layout

```
// Root layout for entire application
// - Sets up fonts (Inter, Manrope)
// - Wraps app in ThemeProvider
// - Defines metadata for SEO
// - Includes global CSS
```

`app/globals.css` - Global Styles

```
/* Global styles including:
   - Tailwind directives
   - CSS custom properties (design tokens)
   - Animation keyframes
   - Utility classes
*/
```

`lib/db.ts` - Prisma Client

```
// Singleton pattern for Prisma client
// Prevents multiple instances in development
// Handles connection pooling
```

`lib/admin-auth.ts` - Authentication

```
// JWT token generation and verification
// Password hashing with bcrypt
// Session management
// Role-based access control
```

`components/ui/*` - shadcn/ui Components

```
// All UI components are copied into the codebase
// NOT installed as npm packages
// Fully customizable
// Based on Radix UI primitives
```

🗄️ DATABASE SCHEMA

Prisma Schema (`prisma/schema.prisma`)

Model: Lead

```
model Lead {
  id                  String   @id @default(uuid())
  email               String   @unique
  name                String
  company             String?
  phone               String?
  confirmationToken   String?
  confirmed           Boolean  @default(false)
  calculatorData      Json?
  source              String?  @default("ai-workforce-calculator")
  createdAt           DateTime @default(now())
  updatedAt           DateTime @updatedAt

  calculatorResponses CalculatorResponse[]
}
```

Purpose: Store all leads captured from forms and calculators

Fields:
- `email` - Unique email (primary identifier)
- `name` - Full name
- `company` - Optional company name
- `phone` - Optional phone number
- `confirmationToken` - Email verification token
- `confirmed` - Email verification status
- `calculatorData` - JSON blob of calculator inputs
- `source` - Where lead came from (e.g., “ai-workforce-calculator”)
- `calculatorResponses` - Related calculator submissions

Indexes:
- `email` - Fast lookup by email
- `confirmationToken` - Fast verification
- `confirmed` - Filter by verification status
- `createdAt` - Sort by newest leads

Model: CalculatorResponse

```
model CalculatorResponse {
  id                 String   @id @default(uuid())
  calculatorType     String
  responses          Json
  recommendations    Json?
  projectedSavings   Json?
  timestamp          DateTime @default(now())
  leadId             String?

  lead               Lead?    @relation(fields: [leadId], references: [id])
}
```

Purpose: Store all calculator submissions with ROI data

Fields:
- `calculatorType` - Which calculator (e.g., "ai-team", "automation")
- `responses` - User's form inputs (JSON)
- `recommendations` - Generated service recommendations (JSON)
- `projectedSavings` - ROI calculations (JSON)
- `leadId` - Foreign key to Lead (optional)

Indexes:
- `calculatorType` + `timestamp` - Filter by calculator type and sort by date

Model: ContactSubmission

```
model ContactSubmission {
  id        String   @id @default(uuid())
  name      String
  email     String
  company   String?
  phone     String?
  subject   String?
  message   String
  status    String   @default("new")
  createdAt DateTime @default(now())
}
```

Purpose: Store general contact form submissions

Fields:
- `status` - "new", "in-progress", "resolved", "closed"

Indexes:
- `createdAt` - Sort by newest
- `status` - Filter by status

Model: AdminUser

```
model AdminUser {
  id           String   @id @default(uuid())
  email        String   @unique
  passwordHash String
  name         String
  role         String   @default("admin")
  createdAt    DateTime @default(now())
  lastLogin    DateTime?
}
```

Purpose: Admin user accounts for dashboard access

Fields:
- `passwordHash` - Bcrypt-hashed password
- `role` - “admin”, “superadmin” (future: role-based permissions)
- `lastLogin` - Track login activity

Indexes:
- `email` - Fast login lookup

Model: BlogApiKey

```
model BlogApiKey {
  id          String   @id @default(uuid())
  keyName     String
  apiKey      String   @unique
  isActive    Boolean  @default(true)
  createdAt   DateTime @default(now())
  lastUsedAt  DateTime?
  usageCount  Int      @default(0)
}
```

Purpose: API keys for blog content management

Fields:
- `isActive` - Enable/disable keys
- `usageCount` - Track API usage

Indexes:
- `apiKey` - Fast API key lookup

Model: ConsultationBooking

```
model ConsultationBooking {
  id              String   @id @default(uuid())
  name            String
  email           String
  company         String?
  phone           String?
  serviceInterest String
  message         String?
  preferredDate   String?
  preferredTime   String?
  status          String   @default("pending")
  googleEventId   String?
  tidycalBookingId String?
  createdAt       DateTime @default(now())
}
```

Purpose: Store consultation booking requests

Fields:
- `serviceInterest` - Which service they're interested in
- `googleEventId` - Linked Google Calendar event
- `tidycalBookingId` - Linked TidyCal booking
- `status` - "pending", "confirmed", "completed", "cancelled"

Indexes:
- `createdAt` - Sort by newest
- `status` - Filter by status

Database Connection

Provider: Supabase (PostgreSQL)

Connection String Format:

```
postgresql://USER:PASSWORD@HOST:PORT/DATABASE?schema=public
```

Environment Variable:

```
DATABASE_URL="postgresql://..."
```

Connection Pooling: Handled by Supabase automatically

Prisma Commands

```
# Generate Prisma client (after schema changes)
npx prisma generate

# Push schema changes to database (dev)
npx prisma db push

# Create migration (production)
npx prisma migrate dev --name migration_name

# Apply migrations (production)
npx prisma migrate deploy

# Open Prisma Studio (database GUI)
npx prisma studio

# Reset database (WARNING: Deletes all data)
npx prisma migrate reset
```

Database Backup

Supabase Automatic Backups:
- Daily automatic backups
- Point-in-time recovery (7 days)
- Manual backup via Supabase dashboard

Manual Backup:

```
# Export database
pg_dump DATABASE_URL > backup_$(date +%Y%m%d).sql

# Restore database
psql DATABASE_URL < backup_20251111.sql
```

🔌 API ENDPOINTS

Public API Routes

1. `/api/breach-check` - Email Breach Checker

Method: POST
Purpose: Check if email has been in data breaches (Have I Been Pwned API)

Request Body:

```
{
  "email": "user@example.com"
}
```

Response:

```
{
  "breached": true,
  "breachCount": 3,
  "breaches": [
    {
      "name": "Adobe",
      "breachDate": "2013-10-04",
      "description": "...",
      "dataClasses": ["Emails", "Passwords"]
    }
  ]
}
```

Rate Limiting: 1500ms delay between requests (HIBP requirement)

Error Handling:
- 400: Invalid email format
- 429: Rate limit exceeded
- 503: HIBP API unavailable

2. `/api/calculator/recommendations` - Calculator Submissions

Method: POST
Purpose: Process calculator submissions, generate recommendations, store data

Request Body:

```
{
  "calculatorType": "ai-team",
  "responses": {
    "employees": 5,
    "avgSalary": 50000,
    "tasks": ["data-entry", "customer-support"]
  },
  "leadData": {
    "name": "John Doe",
    "email": "john@example.com",
    "company": "Acme Inc"
  }
}
```

Response:

```
{
  "success": true,
  "recommendations": [
    {
      "service": "AI Workforce",
      "reason": "High employee replacement potential",
      "estimatedROI": "400%"
    }
  ],
  "projectedSavings": {
    "monthly": 15000,
    "annual": 180000,
    "fiveYear": 900000
  }
}
```

Side Effects:
- Creates/updates Lead record
- Creates CalculatorResponse record
- Sends confirmation email
- Sends Discord notification

3. `/api/consultation` - Consultation Booking

Method: POST
Purpose: Book consultation appointments

Request Body:

```
{
  "name": "Jane Smith",
  "email": "jane@example.com",
  "company": "Tech Corp",
  "phone": "+1-555-0100",
  "serviceInterest": "AI Workforce",
  "message": "Interested in automating customer support",
  "preferredDate": "2025-11-15",
  "preferredTime": "10:00 AM"
}
```

Response:

```
{
  "success": true,
  "bookingId": "uuid",
  "googleEventId": "google_event_id",
  "tidycalBookingId": "tidycal_booking_id"
}
```

Side Effects:
- Creates ConsultationBooking record
- Creates Google Calendar event
- Creates TidyCal booking
- Sends confirmation email
- Sends Discord notification

4. `/api/contact-support` - Contact Form

Method: POST
Purpose: General contact form submissions

Request Body:

```json
{
  "name": "Bob Johnson",
  "email": "bob@example.com",
  "company": "StartupCo",
  "phone": "+1-555-0200",
  "subject": "Pricing Question",
  "message": "How much does AI Workforce cost?"
}
```

Response:

```json
{
  "success": true,
  "submissionId": "uuid"
}
```

Side Effects:
- Creates ContactSubmission record
- Sends notification email
- Sends Discord notification

5. `/api/lead-magnet` - Lead Magnet Downloads

Method: POST
Purpose: Capture leads for downloadable resources

Request Body:

```json
{
  "name": "Sarah Lee",
  "email": "sarah@example.com",
  "company": "GrowthCo",
  "magnetType": "automation-guide"
}
```

Response:

```json
{
  "success": true,
  "downloadUrl": "https://..."
}
```

Side Effects:
- Creates Lead record
- Sends email with download link
- Sends Discord notification

6. `/api/automation-lead` - Automation Page Lead Capture

Method: POST
Purpose: Capture leads from automation service page

Request Body:

```
{
  "name": "Mike Chen",
  "email": "mike@example.com",
  "company": "AutoCo",
  "interest": "Process automation"
}
```

Response:

```
{
  "success": true,
  "leadId": "uuid"
}
```

7. `/api/automation-calc` - Automation Calculator

Method: POST
Purpose: Automation-specific calculator

Request Body:

```
{
  "processHours": 40,
  "employeeCount": 10,
  "avgHourlyRate": 25
}
```

Response:

```
{
  "success": true,
  "savings": {
    "monthly": 10000,
    "annual": 120000,
    "timeRecovered": "400 hours/month"
  }
}
```

8. `/api/tidycal-webhook` - TidyCal Webhook

Method: POST
Purpose: Receive booking notifications from TidyCal

Request Body: (TidyCal format)

```json
{
  "event": "booking.created",
  "data": {
    "bookingId": "...",
    "email": "...",
    "name": "...",
    "datetime": "..."
  }
}
```

Response:

```json
{
  "received": true
}
```

Side Effects:
- Updates ConsultationBooking record
- Sends Discord notification

Admin API Routes (Protected)

Authentication Required

All admin routes require JWT token in Authorization header:

```
Authorization: Bearer <jwt_token>
```

1. `/api/admin/auth/login` - Admin Login

Method: POST
Purpose: Authenticate admin users

Request Body:

```json
{
  "email": "admin@quantumleap.io",
  "password": "secure_password"
}
```

Response:

```json
{
  "success": true,
  "token": "jwt_token",
  "user": {
    "id": "uuid",
    "email": "admin@quantumleap.io",
    "name": "Admin User",
    "role": "admin"
  }
}
```

Sets HTTP-Only Cookie: `admin_token`

2. `/api/admin/auth/verify` - Verify Session

Method: GET
Purpose: Verify JWT token is valid

Response:

```json
{
  "valid": true,
  "user": {
    "id": "uuid",
    "email": "admin@quantumleap.io",
    "name": "Admin User"
  }
}
```

3. `/api/admin/leads` - Lead Management

Method: GET
Purpose: Retrieve all leads with pagination

Query Parameters:
- `page` - Page number (default: 1)
- `limit` - Results per page (default: 50)
- `status` - Filter by status (“confirmed”, “unconfirmed”, “all”)
- `source` - Filter by source

Response:

```json
{
  "leads": [
    {
      "id": "uuid",
      "name": "John Doe",
      "email": "john@example.com",
      "company": "Acme Inc",
      "source": "ai-team-calculator",
      "confirmed": true,
      "createdAt": "2025-11-11T10:00:00Z"
    }
  ],
  "total": 150,
  "page": 1,
  "pages": 3
}
```

Method: PATCH
Purpose: Update lead status

Request Body:

```json
{
  "leadId": "uuid",
  "status": "contacted"
}
```

4. `/api/admin/consultations` - Consultation Management

Method: GET
Purpose: Retrieve all consultation bookings

Response:

```json
{
  "bookings": [
    {
      "id": "uuid",
      "name": "Jane Smith",
      "email": "jane@example.com",
      "serviceInterest": "AI Workforce",
      "preferredDate": "2025-11-15",
      "status": "pending",
      "createdAt": "2025-11-11T10:00:00Z"
    }
  ]
}
```

5. `/api/admin/analytics` - Analytics Dashboard

Method: GET
Purpose: Retrieve analytics data

Response:

```json
{
  "summary": {
    "totalLeads": 150,
    "newLeadsToday": 5,
    "consultationsBooked": 20,
    "calculatorCompletions": 75
  },
  "leadsBySource": {
    "ai-team-calculator": 50,
    "automation-calculator": 40,
    "contact-form": 30,
    "lead-magnet": 30
  },
  "conversionRates": {
    "calculatorToLead": 0.45,
    "leadToConsultation": 0.13
  }
}
```

6. `/api/admin/calendar` - Calendar Integration

Method: GET
Purpose: Fetch Google Calendar events

Response:

```json
{
  "events": [
    {
      "id": "google_event_id",
      "summary": "Consultation with Jane Smith",
      "start": "2025-11-15T10:00:00Z",
      "attendees": ["jane@example.com"]
    }
  ]
}
```

Method: POST
Purpose: Create calendar event

Request Body:

```json
{
  "summary": "Consultation with John Doe",
  "description": "AI Workforce consultation",
  "start": "2025-11-16T14:00:00Z",
  "end": "2025-11-16T15:00:00Z",
  "attendees": ["john@example.com"]
}
```

7. `/api/admin/blog-api-keys` - Blog API Key Management

Method: GET
Purpose: List all blog API keys

Method: POST
Purpose: Create new API key

Method: DELETE
Purpose: Revoke API key

8. `/api/admin/google-drive` - Google Drive Integration

Method: GET
Purpose: List files in Google Drive

Method: POST
Purpose: Upload file to Google Drive

API Error Responses

Standard Error Format:

```json
{
  "error": true,
  "message": "Descriptive error message",
  "code": "ERROR_CODE",
  "details": {} // Optional additional details
}
```

Common Error Codes:
- `VALIDATION_ERROR` - Invalid input data
- `UNAUTHORIZED` - Missing or invalid authentication

- `FORBIDDEN` - Insufficient permissions
- `NOT_FOUND` - Resource not found
- `RATE_LIMIT_EXCEEDED` - Too many requests
- `INTERNAL_ERROR` - Server error

HTTP Status Codes:
- `200` - Success
- `201` - Created
- `400` - Bad Request
- `401` - Unauthorized
- `403` - Forbidden
- `404` - Not Found
- `429` - Too Many Requests
- `500` - Internal Server Error
- `503` - Service Unavailable

🔗 THIRD-PARTY INTEGRATIONS

1. Gmail API (Email Sending)

Purpose: Send transactional emails (confirmations, notifications)

Authentication: OAuth 2.0 Service Account

Setup:
1. Create Google Cloud Project
2. Enable Gmail API
3. Create Service Account
4. Download JSON key
5. Grant domain-wide delegation
6. Add credentials to `.env`

Environment Variables:

```
GMAIL_CLIENT_EMAIL=service-account@project.iam.gserviceaccount.com
GMAIL_PRIVATE_KEY="-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----\n"
GMAIL_SENDER=paras@leapintoai.com
```

Implementation: `lib/email.ts`

Usage:

```
import { sendEmail } from '@/lib/email'

await sendEmail({
  to: 'user@example.com',
  subject: 'Consultation Confirmation',
  html: '<h1>Thank you for booking!</h1>',
  text: 'Thank you for booking!'
})
```

Rate Limits:
- 100 emails/second
- 1,000 emails/day (free tier)
- 10,000 emails/day (paid)

Error Handling:
- Retry failed sends up to 3 times
- Log all email errors
- Send alert to Discord on failures

2. Google Calendar API (Booking Management)

Purpose: Create and manage consultation bookings

Authentication: OAuth 2.0 Service Account

Environment Variables:

```
GOOGLE_CALENDAR_ID=primary
# Uses same service account as Gmail
```

Implementation: `lib/google-calendar.ts`

Usage:

```
import { createCalendarEvent } from '@/lib/google-calendar'

const eventId = await createCalendarEvent({
  summary: 'Consultation with John Doe',
  description: 'AI Workforce consultation',
  start: '2025-11-15T10:00:00Z',
  end: '2025-11-15T11:00:00Z',
  attendees: ['john@example.com']
})
```

Features:
- Create events
- Update events
- Delete events
- List events
- Send email invitations
- Set reminders

3. TidyCal API (Alternative Booking)

Purpose: Alternative booking system with scheduling widget

Authentication: API Key

Environment Variables:

```
TIDYCAL_API_KEY=your_api_key_here
NEXT_PUBLIC_TIDYCAL_BOOK_URL=https://tidycal.com/quantumleap/consultation
```

Implementation: `lib/tidycal.ts`

Usage:

```
import { createTidycalBooking } from '@/lib/tidycal'

const bookingId = await createTidycalBooking({
  name: 'John Doe',
  email: 'john@example.com',
  datetime: '2025-11-15T10:00:00Z'
})
```

Webhook: `/api/tidycal-webhook` receives booking notifications

Features:
- Embedded booking widget
- Automatic calendar sync
- Reminder emails
- Timezone handling
- Payment processing (optional)

4. Have I Been Pwned (HIBP) API (Breach Checking)

Purpose: Check if emails have been in data breaches

Authentication: API Key (optional for higher rate limits)

API Endpoint: `https://haveibeenpwned.com/api/v3/breachedaccount/{email}`

Implementation: `app/api/breach-check/route.ts`

Rate Limiting:
- Free: 1 request per 1500ms
- Paid: Higher limits

Privacy:
- No email storage (k-Anonymity model)
- Hashed email lookups
- GDPR compliant

Usage:

```
const response = await fetch(
  `https://haveibeenpwned.com/api/v3/breachedaccount/${email}`,
  {
    headers: {
      'hibp-api-key': process.env.HIBP_API_KEY,
      'user-agent': 'QuantumLeap-BreachChecker'
    }
  }
)
```

5. Discord Webhooks (Internal Notifications)

Purpose: Real-time notifications for new leads and bookings

Authentication: Webhook URL (secret)

Environment Variables:

```
DISCORD_WEBHOOK_URL=https://discord.com/api/webhooks/...
```

Implementation: `lib/discord.ts`

Usage:

```
import { sendDiscordNotification } from '@/lib/discord'

await sendDiscordNotification({
  title: '🎯 New Lead Captured',
  description: 'John Doe from Acme Inc',
  fields: [
    { name: 'Email', value: 'john@example.com' },
    { name: 'Source', value: 'AI Team Calculator' }
  ],
  color: 0x7c3aed // Purple
})
```

Notification Types:
- New lead captured
- Calculator completion
- Consultation booking
- Contact form submission
- API errors

6. Firebase (Storage & Future Auth)

Purpose: File storage and potential authentication

Authentication: Service Account JSON

Environment Variables:

```
FIREBASE_PROJECT_ID=your-project-id
FIREBASE_CLIENT_EMAIL=firebase-adminsdk@your-project-id.iam.gserviceaccount.com
FIREBASE_PRIVATE_KEY="-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----\n"
```

Implementation: `lib/firebase-admin.ts`

Current Use: File storage for blog images

Future Use: User authentication, real-time database

7. Supabase (Database & Future Features)

Purpose: PostgreSQL database hosting

Authentication: Anon Key (public) + Service Role Key (private)

Environment Variables:

```
NEXT_PUBLIC_SUPABASE_URL=https://your-project.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=your-anon-key
SUPABASE_SERVICE_ROLE_KEY=your-service-role-key (if needed)
```

Implementation: `lib/supabase.ts`

Current Use: Database only (via Prisma)

Future Use:
- Authentication
- Real-time subscriptions
- File storage
- Edge Functions

⚙️ ENVIRONMENT CONFIGURATION

Required Environment Variables

Create `.env.local` file in `nextjs_space/` directory:

```
# ============================================
# DATABASE
# ============================================
DATABASE_URL="postgresql://USER:PASSWORD@HOST:PORT/DATABASE?schema=public"

# Get from: /home/ubuntu/quantumleap_credentials.json
# Supabase connection string

# ============================================
# GMAIL API (Email Sending)
# ============================================
GMAIL_CLIENT_EMAIL="service-account@project.iam.gserviceaccount.com"
GMAIL_PRIVATE_KEY="-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----\n"
GMAIL_SENDER="paras@leapintoai.com"

# ============================================
# GOOGLE CALENDAR API
# ============================================
GOOGLE_CALENDAR_ID="primary"
# Uses same service account as Gmail

# ============================================
# TIDYCAL API
# ============================================
TIDYCAL_API_KEY="your_tidycal_api_key"
NEXT_PUBLIC_TIDYCAL_BOOK_URL="https://tidycal.com/quantumleap/consultation"

# ============================================
# DISCORD WEBHOOKS
# ============================================
DISCORD_WEBHOOK_URL="https://discord.com/api/webhooks/1431797531377143818/..."

# Get from: /home/ubuntu/quantumleap_credentials.json

# ============================================
# FIREBASE (Storage)
# ============================================
FIREBASE_PROJECT_ID="your-firebase-project-id"
FIREBASE_CLIENT_EMAIL="firebase-adminsdk@your-project-id.iam.gserviceaccount.com"
FIREBASE_PRIVATE_KEY="-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----\n"

# ============================================
# SUPABASE (Database)
# ============================================
NEXT_PUBLIC_SUPABASE_URL="https://ilvxbllftqzewiojsryq.supabase.co"
NEXT_PUBLIC_SUPABASE_ANON_KEY="eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."

# Get from: /home/ubuntu/quantumleap_credentials.json

# ============================================
# ADMIN AUTHENTICATION
# ============================================
ADMIN_JWT_SECRET="your-super-secret-jwt-key-change-this-in-production"

# Generate with: openssl rand -base64 32

# ============================================
# HAVE I BEEN PWNED API (Optional)
# ============================================
HIBP_API_KEY="your-hibp-api-key"

# Get from: https://haveibeenpwned.com/API/Key
# Optional - increases rate limits

# ============================================
# GOOGLE DRIVE (Optional)
# ============================================
GOOGLE_DRIVE_FOLDER_ID="your-folder-id"

# Uses same service account as Gmail

# ============================================
# APPLICATION SETTINGS
# ============================================
NEXT_PUBLIC_BASE_URL="https://quantumleapai.abacusai.app"
NODE_ENV="production"

# For local development:
# NEXT_PUBLIC_BASE_URL="http://localhost:3000"
# NODE_ENV="development"

# ============================================
# HOSTINGER SMTP (Alternative Email)
# ============================================
SMTP_HOST="smtp.hostinger.com"
SMTP_PORT="587"
SMTP_USER="paras@leapintoai.com"
SMTP_PASSWORD="Walle2026."
SMTP_FROM_NAME="QuantumLeap AI"

# Get from: /home/ubuntu/quantumleap_credentials.json
# Note: Currently using Gmail API, but SMTP is configured as backup
```

Getting Credentials

All credentials are stored in:

```
/home/ubuntu/quantumleap_credentials.json
```

To view credentials:

```
cat /home/ubuntu/quantumleap_credentials.json
```

Copy to your local environment:

```
# On server
cat /home/ubuntu/quantumleap_credentials.json

# Copy output to your local machine
# Then create .env.local and paste values
```

Environment Variable Priority

1. `.env.local` - Local overrides (gitignored)
2. `.env.production` - Production defaults
3. `.env.development` - Development defaults
4. `.env` - Default values

Never commit:
- `.env.local`
- `.env.production.local`
- `.env.development.local`

Can commit:
- `.env.example` - Template with placeholder values

Verifying Environment Variables

Check if all required variables are set:

```
cd nextjs_space
node -e "
const requiredVars = [
  'DATABASE_URL',
  'GMAIL_CLIENT_EMAIL',
  'GMAIL_PRIVATE_KEY',
  'DISCORD_WEBHOOK_URL',
  'ADMIN_JWT_SECRET'
];
requiredVars.forEach(v => {
  console.log(v + ':', process.env[v] ? '✅' : '❌');
});
"
```

Test database connection:

```
npx prisma studio
# Should open browser with database GUI
```

🧩 KEY COMPONENTS

1. FlameBorder Component (`components/ui/flame-border.tsx`)

Purpose: Animated copper flame border effect for cards

Usage:

```
import { FlameBorder } from '@/components/ui/flame-border'

<FlameBorder className="relative">
  <div className="p-6">
    Card content here
  </div>
</FlameBorder>
```

Features:
- SVG-based animation
- 28-second cycle
- Copper color (#b87333)
- Thickness: 3px
- Glow effect with mix-blend-screen

Props:
- `className` - Additional CSS classes
- `children` - Card content

2. Calculator Components (`components/calculator/`)

AITeamCalculator

File: `ai-team-calculator.tsx`

Purpose: Calculate ROI for replacing employees with AI

Inputs:
- Number of employees
- Average salary
- Tasks performed
- Hours per week

Outputs:
- Annual savings
- 5-year ROI
- Recommended services

Features:
- Real-time calculations
- Service recommendations
- Lead capture form
- Email automation

AutomationCalculator

File: `automation-calculator.tsx`

Purpose: Calculate ROI for process automation

Inputs:
- Process hours per week
- Number of employees
- Average hourly rate
- Process type

Outputs:
- Monthly time savings
- Annual cost savings
- Productivity gains

HiringRiskCalculator

File: `hiring-risk-calculator.tsx`

Purpose: Calculate cost of bad hires

Inputs:
- Position level
- Salary
- Hiring process cost
- Time to productivity

Outputs:
- Total bad hire cost
- Risk assessment
- Recommended background check level

ProfitPotentialCalculator

File: `profit-potential-calculator.tsx`

Purpose: Calculate business transformation ROI

Inputs:
- Current revenue
- Current profit margin
- Industry
- Growth goals

Outputs:
- Potential profit increase
- Digital transformation ROI
- Recommended services

3. Exit Intent System

ExitIntentProvider

File: `components/exit-intent-provider.tsx`

Purpose: Detect user exit intent and show popup

Triggers:
- Mouse leaves top of page
- Scroll depth > 50%
- Idle time > 30 seconds

Configuration:

```
<ExitIntentProvider
  enableMouse={true}
  enableScroll={true}
  enableIdle={true}
  idleTime={30000}
>
  {children}
</ExitIntentProvider>
```

Page-Specific Exit Intents

- `automation-exit-intent.tsx` - Automation page popup
- `background-checks-exit-intent.tsx` - Background checks popup
- `business-transformation-exit-intent.tsx` - Transformation popup

Features:
- Session storage (one popup per session)
- A/B testing support
- GA4 event tracking
- Lead capture integration

4. Hero Sections

CoralReefHero

File: `components/sections/coral-reef-hero.tsx`

Purpose: Homepage hero with video and dual CTAs

Features:
- Split-screen layout
- Video autoplay
- Smooth panel transitions
- Trust indicators
- Dual CTA buttons

CTAs:
- "View Small Business Solutions" → `/smb`
- "Stop the Burnout" → `/smb`

Split Hero (Legacy)

File: `components/sections/split-hero-section.tsx`

Status: Deprecated (replaced by CoralReefHero)

5. Advisory Board Carousel

File: `components/sections/advisory-board-section.tsx`

Purpose: Display strategic advisors in rotating carousel

Features:
- Auto-rotation (6 seconds)
- Pause on hover
- 2 cards per view
- Spring physics animations
- Circular profile images

Board Members:
1. Andrea Morris - Product & Audience Innovation
2. Guneet Chhabra - Digital Transformation
3. Jasrita Dhir - Brand & Marketing
4. Karen Mousseau - Talent Acquisition
5. Leonard Kuek - AI Talent & Automation
6. Marlon Gibbs - Talent Acquisition & Recruitment
7. Neil Dhawan, CPA - Finance & Business Transformation
8. Senthil Parthiban - Data & Business Intelligence
9. Dr. Sukhi Pritam - Healthcare & Technology
10. Tapan Jha - Cybersecurity

6. Testimonials Carousel

File: `components/sections/home-testimonials-carousel.tsx`

Purpose: Display client testimonials with smooth animations

Features:
- Swiper.js integration
- Auto-rotation (6 seconds)
- Custom pagination with glow effect
- Before/After indicators
- Star ratings
- Category badges

Testimonials: 5+ real client stories

7. Layout Components

Header

File: `components/layout/header.tsx`

Features:
- Sticky navigation
- Services dropdown
- Mobile menu
- Dark/light toggle
- Logo with link
- CTA button

Navigation:
- Home
- About Us
- Services (dropdown)
- AI Workforce
- Intelligent Automation
- Cyber Intelligence
- Beyond Background Checks
- Business Transformation
- Blog
- Contact

Footer

File: `components/layout/footer.tsx`

Features:
- Newsletter signup
- Social links
- Trust indicators
- Legal links
- Company info

Sections:
- Services
- Company
- Legal
- Contact
- Newsletter

8. SMB Components (`components/smb/`)

ThemeContext

File: `theme-context.tsx`

Purpose: Manage dark/light mode state

Features:
- localStorage persistence
- Default to dark mode
- Document class management
- React context API

AnimatedButton

File: `animated-button.tsx`

Purpose: CTA buttons with ripple effect

Features:
- Ripple animation on click
- Scale hover effect
- Two variants (primary/secondary)
- Accessibility focus states

ScrollProgress

File: `scroll-progress.tsx`

Purpose: Scroll progress indicator

Features:
- Gradient coral-teal bar
- Spring animation
- Fixed position
- Framer Motion integration

AnimatedBackground

File: `animated-background.tsx`

Purpose: Canvas particle system

Features:
- Floating coral/teal shapes
- Parallax effect
- Responsive canvas sizing
- Motion blur transitions
- 60fps rendering

📄 PAGES AND ROUTES

Public Pages (18 Pages)

1. Homepage (`/` - `app/page.tsx`)

Purpose: Main landing page

Sections:
- CoralReef Hero (video + dual CTAs)
- Trust Bar (company logos)
- Value Proposition (split-screen SMB/Enterprise)
- AI Employees Showcase
- Problem Section (manual tasks burden)
- Calculator Section
- Testimonials Carousel
- FAQ Section
- CTA Section

Key Features:
- Exit intent popup
- Breach checker widget
- Service previews
- Social proof

2. About Us (`/about-us` - `app/about-us/page.tsx`)

Purpose: Company information and credentials

Sections:
- Hero (Elite Expertise)
- Founder’s Vision (Paras Khurana bio)
- Elite Cybersecurity Credentials
- Strategic Advisory Board Carousel
- CTA (Book Consultation)

Key Features:
- Founder story with metrics
- NASA recognition
- Board member profiles
- Scroll animations

3. SMB Landing Page (`/smb` - `app/smb/page.tsx`)

Purpose: High-converting page for SMBs

Sections:
- Award-Winning Hero
- Trust Badges
- Service Cards (5 services)
- Testimonials Carousel
- FAQ
- Urgency CTA

Key Features:
- Light mode color corrections
- Theme toggle
- Standardized service cards
- SEO schema markup
- Video background

4. AI Workforce (`/ai-workforce` - `app/ai-workforce/page.tsx`)

Purpose: AI employee replacement service

Sections:
- Hero (Replace Employees with AI)
- AI Team Calculator
- Service Details
- ROI Examples
- Guarantee Section
- Testimonials
- FAQ
- CTA

Key Features:
- Interactive calculator
- Service recommendations
- Real-time ROI calculations
- Lead capture

5. Intelligent Automation (`/intelligent-automation` - `app/intelligent-automation/page.tsx`)

Purpose: Process automation service

Sections:
- Hero Video
- Automation Calculator
- Comparison Table (Manual vs Automated)
- Service Details
- Strategic Blog Integration
- CTA

Key Features:
- Automation ROI calculator
- Exit intent popup
- Process comparison
- Industry examples

6. Cyber Intelligence (`/cyber-intelligence` - `app/cyber-intelligence/page.tsx`)

Purpose: Threat detection and monitoring

Sections:
- Hero (Advanced Threat Detection)
- Service Details
- Threat Landscape
- Security Features

- Testimonials
- FAQ
- CTA

Key Features:
- Security breach examples
- Real-time monitoring details
- Compliance information
- Enterprise focus

7. Beyond Background Checks (`/background-checks` - `app/background-checks/page.tsx`)

Purpose: Deep background investigation service

Sections:
- Hero (Prevent Bad Hires)
- Hiring Risk Calculator
- Service Details
- Investigation Process
- Guarantee Section
- Testimonials Carousel
- Strategic Blog Section
- FAQ (expanded)
- CTA

Key Features:
- Bad hire cost calculator
- Real-world case studies
- ROI math ($5K investigation vs $200K-4M bad hire)
- Exit intent popup

8. Business Transformation (`/business-transformation` - `app/business-transformation/page.tsx`)

Purpose: Full digital transformation service

Sections:
- Hero (Transform Your Business)
- Profit Potential Calculator
- Transformation Process
- Founder Testimonials with Metrics
- Service Details
- Strategic Blog Integration
- CTA

Key Features:
- Profit potential calculator
- ROI projections
- Industry examples
- Exit intent popup

9. Blog Listing (`/blog` - `app/blog/page.tsx`)

Purpose: List all blog posts

Features:
- Grid layout
- Category filters
- Search functionality
- Pagination
- SEO-optimized

10. Blog Post (`/blog/[slug]` - `app/blog/[slug]/page.tsx`)

Purpose: Individual blog post

Sections:
- Hero (post title/image)
- Content (markdown/HTML)
- Related Posts
- CTA Section

Key Features:
- Dynamic content loading
- SEO optimization
- Internal linking
- Social sharing

11. Consultation Booking (`/consultation` - `app/consultation/page.tsx`)

Purpose: Book consultation appointments

Form Fields:
- Name
- Email
- Company
- Phone
- Service Interest (dropdown)
- Preferred Date/Time
- Message

Features:
- Date/time picker
- Service dropdown
- TidyCal widget integration
- Google Calendar integration
- Confirmation email

12. Consultation Contact (`/intelligent-automation/contact` - `app/intelligent-automation/contact/page.tsx`)

Purpose: Contact form specific to automation

Features:
- Pre-filled service interest
- Simplified form
- Direct to consultation flow

13. Confirmation Page (`/confirmation` - `app/confirmation/page.tsx`)

Purpose: Booking confirmation

Features:
- Success message
- Next steps
- Calendar add link
- Social sharing

14. Privacy Policy (`/privacy` - `app/privacy/page.tsx`)

Purpose: Legal privacy policy

Sections:
- Data collection
- Data usage
- Data protection
- User rights
- Contact info

15. Terms of Service (`/terms` - `app/terms/page.tsx`)

Purpose: Legal terms and conditions

Sections:
- Service terms
- User responsibilities
- Liability limitations
- Dispute resolution
- Contact info

16. Coral Reef Demo (`/coral-reef-demo` - `app/coral-reef-demo/page.tsx`)

Purpose: Design system showcase

Features:
- Color swatches
- Typography examples
- Component demos
- Live preview

Status: Internal use only (not linked in nav)

Admin Pages (5 Pages)

Protected by JWT Authentication

17. Admin Login (`/admin/login` - `app/admin/login/page.tsx`)

Purpose: Admin authentication

Form Fields:
- Email
- Password

Features:
- JWT token generation
- HTTP-only cookie
- Session management
- Redirect to dashboard

18. Admin Dashboard (`/admin` - `app/admin/page.tsx`)

Purpose: Overview of all admin functions

Sections:
- Analytics Summary
- Total Leads
- New Leads Today
- Consultations Booked
- Calculator Completions
- Quick Actions
- View Leads
- Manage Consultations
- Calendar
- CRM
- Recent Activity

Features:
- Real-time metrics
- Quick action buttons
- Recent activity feed

19. Lead Management (`/admin/leads` - `app/admin/leads/page.tsx`)

Purpose: Manage all leads

Features:
- Sortable table
- Filter by source/status
- Search by name/email
- Bulk actions
- Export to CSV
- Lead details modal

Columns:
- Name
- Email
- Company
- Source
- Confirmed
- Created At
- Actions

20. CRM (`/admin/crm` - `app/admin/crm/page.tsx`)

Purpose: Customer relationship management

Features:
- Lead pipeline view
- Status management
- Notes/comments
- Task assignments
- Follow-up reminders

21. Calendar Management (`/admin/calendar` - `app/admin/calendar/page.tsx`)

Purpose: Manage consultation bookings

Features:
- Calendar view (day/week/month)
- Google Calendar sync
- TidyCal integration
- Drag-and-drop rescheduling
- Booking details

22. TidyCal Integration (`/admin/tidycal` - `app/admin/tidycal/page.tsx`)

Purpose: Manage TidyCal settings

Features:
- Webhook configuration
- Booking settings
- Availability management
- Integration status

23. Blog API Management (`/admin/blog-api` - `app/admin/blog-api-keys/page.tsx`)

Purpose: Manage blog API keys

Features:
- Create new keys
- Revoke keys
- View usage stats
- Key permissions

Route Protection

Public Routes: No authentication required

Admin Routes: JWT authentication required

Authentication Check:

```
// middleware.ts
import { NextResponse } from 'next/server'
import type { NextRequest } from 'next/server'
import { verifyToken } from '@/lib/admin-auth'

export async function middleware(request: NextRequest) {
  const { pathname } = request.nextUrl

  // Protect /admin routes (except /admin/login)
  if (pathname.startsWith('/admin') && pathname !== '/admin/login') {
    const token = request.cookies.get('admin_token')?.value

    if (!token) {
      return NextResponse.redirect(new URL('/admin/login', request.url))
    }

    try {
      await verifyToken(token)
      return NextResponse.next()
    } catch (error) {
      return NextResponse.redirect(new URL('/admin/login', request.url))
    }
  }

  return NextResponse.next()
}

export const config = {
  matcher: '/admin/:path*'
}
```

🔐 AUTHENTICATION SYSTEM

Admin Authentication

Type: JWT-based authentication

Flow:
1. Admin enters credentials on `/admin/login`
2. POST to `/api/admin/auth/login`
3. Verify email/password against `admin_users` table
4. Generate JWT token with payload:
`json`
```
  {
    "userId": "uuid",
    "email": "admin@quantumleap.io",
    "role": "admin",
    "iat": 1699000000,
    "exp": 1699086400
  }
```
5. Set HTTP-only cookie: `admin_token`
6. Return user data to frontend
7. Redirect to `/admin`

Token Expiration: 24 hours

Token Refresh: Currently manual (logout/login)

Security Features:
- HTTP-only cookies (not accessible via JavaScript)
- Secure flag (HTTPS only in production)
- SameSite=Lax (CSRF protection)
- Password hashing with bcrypt (10 rounds)

Implementation

File: `lib/admin-auth.ts`

```
import bcrypt from 'bcryptjs'
import jwt from 'jsonwebtoken'

const JWT_SECRET = process.env.ADMIN_JWT_SECRET!

export async function hashPassword(password: string): Promise<string> {
  return bcrypt.hash(password, 10)
}

export async function verifyPassword(
  password: string,
  hashedPassword: string
): Promise<boolean> {
  return bcrypt.compare(password, hashedPassword)
}

export function generateToken(payload: object): string {
  return jwt.sign(payload, JWT_SECRET, {
    expiresIn: '24h'
  })
}

export function verifyToken(token: string): any {
  try {
    return jwt.verify(token, JWT_SECRET)
  } catch (error) {
    throw new Error('Invalid token')
  }
}
```

Login API

File: `app/api/admin/auth/login/route.ts`

```typescript
import { NextRequest, NextResponse } from 'next/server'
import { db } from '@/lib/db'
import { verifyPassword, generateToken } from '@/lib/admin-auth'

export async function POST(request: NextRequest) {
  const { email, password } = await request.json()

  // Find admin user
  const admin = await db.adminUser.findUnique({
    where: { email }
  })

  if (!admin) {
    return NextResponse.json(
      { error: 'Invalid credentials' },
      { status: 401 }
    )
  }

  // Verify password
  const valid = await verifyPassword(password, admin.passwordHash)

  if (!valid) {
    return NextResponse.json(
      { error: 'Invalid credentials' },
      { status: 401 }
    )
  }

  // Generate token
  const token = generateToken({
    userId: admin.id,
    email: admin.email,
    role: admin.role
  })

  // Update last login
  await db.adminUser.update({
    where: { id: admin.id },
    data: { lastLogin: new Date() }
  })

  // Set HTTP-only cookie
  const response = NextResponse.json({
    success: true,
    user: {
      id: admin.id,
      email: admin.email,
      name: admin.name,
      role: admin.role
    }
  })

  response.cookies.set('admin_token', token, {
    httpOnly: true,
    secure: process.env.NODE_ENV === 'production',
    sameSite: 'lax',
    maxAge: 86400 // 24 hours
  })

  return response
}
```

Protected Route Example

File: `app/admin/leads/page.tsx`

```
'use client'

import { useEffect, useState } from 'react'
import { useRouter } from 'next/navigation'

export default function LeadsPage() {
  const router = useRouter()
  const [leads, setLeads] = useState([])
  const [loading, setLoading] = useState(true)

  useEffect(() => {
    // Verify authentication
    fetch('/api/admin/auth/verify')
      .then(res => {
        if (!res.ok) {
          router.push('/admin/login')
        }
        return res.json()
      })
      .then(() => {
        // Fetch leads
        return fetch('/api/admin/leads')
      })
      .then(res => res.json())
      .then(data => {
        setLeads(data.leads)
        setLoading(false)
      })
      .catch(() => {
        router.push('/admin/login')
      })
  }, [router])

  // ... rest of component
}
```

Creating Admin Users

Script: `scripts/init-admin.ts`

```
import { db } from '@/lib/db'
import { hashPassword } from '@/lib/admin-auth'

async function createAdmin() {
  const email = 'admin@quantumleap.io'
  const password = 'SecurePassword123!'
  const name = 'Admin User'

  const passwordHash = await hashPassword(password)

  const admin = await db.adminUser.create({
    data: {
      email,
      passwordHash,
      name,
      role: 'admin'
    }
  })

  console.log('Admin user created:', admin.email)
}

createAdmin()
```

Run script:

```
npx tsx scripts/init-admin.ts
```

Security Best Practices

1. **Never expose JWT_SECRET**
   - Keep in `.env.local`
   - Use strong random string (32+ chars)
   - Rotate periodically

2. **Use HTTP-only cookies**
   - Prevents XSS attacks
   - Not accessible via JavaScript

3. **Hash passwords properly**
   - Use bcrypt with 10+ rounds
   - Never store plaintext

4. **Implement rate limiting**
   - Prevent brute force attacks
   - Limit login attempts

5. **Use HTTPS in production**
   - Secure cookie flag
   - Prevents MITM attacks

6. **Implement token refresh**
   - Reduce token lifetime
   - Implement refresh token flow

📧 EMAIL SYSTEM

Email Provider

Current: Gmail API (OAuth 2.0 Service Account)

Backup: Hostinger SMTP (configured but not actively used)

Gmail API Setup

Authentication: Service Account with domain-wide delegation

Configuration:

```
GMAIL_CLIENT_EMAIL=service-account@project.iam.gserviceaccount.com
GMAIL_PRIVATE_KEY="-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----\n"
GMAIL_SENDER=paras@leapintoai.com
```

Implementation: `lib/email.ts`

```
import { google } from 'googleapis'

const gmail = google.gmail({
  version: 'v1',
  auth: new google.auth.JWT(
    process.env.GMAIL_CLIENT_EMAIL,
    undefined,
    process.env.GMAIL_PRIVATE_KEY?.replace(/\\n/g, '\n'),
    ['https://www.googleapis.com/auth/gmail.send']
  )
})

export async function sendEmail({
  to,
  subject,
  html,
  text
}: {
  to: string
  subject: string
  html: string
  text: string
}) {
  const message = [
    `From: QuantumLeap AI <${process.env.GMAIL_SENDER}>`,
    `To: ${to}`,
    `Subject: ${subject}`,
    'MIME-Version: 1.0',
    'Content-Type: text/html; charset=utf-8',
    '',
    html
  ].join('\n')

  const encodedMessage = Buffer.from(message)
    .toString('base64')
    .replace(/\+/g, '-')
    .replace(/\//g, '_')
    .replace(/=+$/, '')

  await gmail.users.messages.send({
    userId: 'me',
    requestBody: {
      raw: encodedMessage
    }
  })
}
```

Email Types

1. Consultation Confirmation

Trigger: User books consultation

Template:

```html
<h1>Consultation Confirmed</h1>
<p>Thank you for booking a consultation with QuantumLeap AI.</p>

<h2>Booking Details:</h2>
<ul>
  <li>Date: {preferredDate}</li>
  <li>Time: {preferredTime}</li>
  <li>Service: {serviceInterest}</li>
</ul>

<h2>What to Expect:</h2>
<p>During your consultation, we'll:</p>
<ul>
  <li>Discuss your specific needs</li>
  <li>Provide tailored recommendations</li>
  <li>Answer all your questions</li>
</ul>

<p>A calendar invitation has been sent separately.</p>

<p>If you need to reschedule, please reply to this email.</p>

<p>Looking forward to speaking with you!</p>
<p>- The QuantumLeap AI Team</p>
```

2. Calculator Results

Trigger: User completes calculator

Template:

```html
<h1>Your ROI Analysis Results</h1>
<p>Thank you for using our {calculatorType} calculator.</p>

<h2>Your Projected Savings:</h2>
<ul>
  <li>Monthly: ${projectedSavings.monthly:,}</li>
  <li>Annual: ${projectedSavings.annual:,}</li>
  <li>5-Year: ${projectedSavings.fiveYear:,}</li>
</ul>

<h2>Recommended Services:</h2>
{#each recommendations}
  <h3>{service}</h3>
  <p>{reason}</p>
  <p>Estimated ROI: {estimatedROI}</p>
{/each}

<p>Ready to get started?</p>
<a href="https://quantumleapai.abacusai.app/consultation">Book a Free Consultation</a>

<p>Questions? Reply to this email and we'll respond within 24 hours.</p>

<p>Best regards,</p>
<p>- The QuantumLeap AI Team</p>
```

3. Lead Magnet Delivery

Trigger: User requests lead magnet (e.g., automation guide)

Template:

```
<h1>Your Free {magnetName} is Ready!</h1>
<p>Thank you for your interest in {magnetName}.</p>

<p>Click below to download your free resource:</p>
<a href="{downloadUrl}">Download {magnetName}</a>

<h2>What's Inside:</h2>
<ul>
  <li>{benefit1}</li>
  <li>{benefit2}</li>
  <li>{benefit3}</li>
</ul>

<p>Want to learn more? Book a free consultation:</p>
<a href="https://quantumleapai.abacusai.app/consultation">Book Free Consultation</a>

<p>Best regards,</p>
<p>- The QuantumLeap AI Team</p>
```

4. Contact Form Confirmation

Trigger: User submits contact form

Template:

```
<h1>We Received Your Message</h1>
<p>Thank you for contacting QuantumLeap AI.</p>

<p>We've received your message and will respond within 24 hours.</p>

<h2>Your Message:</h2>
<p><strong>Subject:</strong> {subject}</p>
<p><strong>Message:</strong> {message}</p>

<p>In the meantime, feel free to explore our services:</p>
<ul>
  <li><a href="https://quantumleapai.abacusai.app/ai-workforce">AI Workforce</a></li>
  <li><a href="https://quantumleapai.abacusai.app/intelligent-automation">Intelligent
Automation</a></li>
  <li><a href="https://quantumleapai.abacusai.app/cyber-intelligence">Cyber Intelli-
gence</a></li>
</ul>

<p>Best regards,</p>
<p>- The QuantumLeap AI Team</p>
```

5. Email Verification (Future)

Trigger: User submits email (optional double opt-in)

Template:

```html
<h1>Verify Your Email</h1>
<p>Thank you for your interest in QuantumLeap AI.</p>

<p>Please verify your email address by clicking below:</p>
<a href="{verificationUrl}">Verify Email</a>

<p>This link expires in 24 hours.</p>

<p>If you didn't request this, please ignore this email.</p>

<p>Best regards,</p>
<p>- The QuantumLeap AI Team</p>
```

Email Automation

File: `lib/email-automation.ts`

Features:
- Queue system for bulk emails
- Retry logic for failed sends
- Template management
- Personalization
- Tracking (opens/clicks - future)

Example:

```typescript
import { sendEmail } from '@/lib/email'
import { generateConsultationConfirmationEmail } from '@/lib/email-templates'

export async function sendConsultationConfirmation(
  booking: ConsultationBooking
) {
  const { html, text } = generateConsultationConfirmationEmail(booking)

  try {
    await sendEmail({
      to: booking.email,
      subject: 'Consultation Confirmed - QuantumLeap AI',
      html,
      text
    })

    console.log('Confirmation email sent to:', booking.email)
  } catch (error) {
    console.error('Failed to send confirmation email:', error)
    // Retry logic or alert admin
  }
}
```

Email Best Practices

1. **Always include unsubscribe link** (for marketing emails)
2. **Use plain text fallback** (for email clients that don't support HTML)
3. **Test emails across clients** (Gmail, Outlook, Apple Mail)
4. **Keep subject lines under 50 characters**
5. **Use clear CTAs** (one primary action per email)

6. **Personalize content** (use recipient's name, company)
7. **Monitor bounce rates** (clean email list regularly)
8. **Track engagement** (opens, clicks - implement in future)

SMTP Backup Configuration

Provider: Hostinger

Configuration:

```
SMTP_HOST=smtp.hostinger.com
SMTP_PORT=587
SMTP_USER=paras@leapintoai.com
SMTP_PASSWORD=Walle2026.
SMTP_FROM_NAME=QuantumLeap AI
```

Usage: (if Gmail API fails)

```
import nodemailer from 'nodemailer'

const transporter = nodemailer.createTransport({
  host: process.env.SMTP_HOST,
  port: parseInt(process.env.SMTP_PORT || '587'),
  secure: false, // TLS
  auth: {
    user: process.env.SMTP_USER,
    pass: process.env.SMTP_PASSWORD
  }
})

await transporter.sendMail({
  from: `"${process.env.SMTP_FROM_NAME}" <${process.env.SMTP_USER}>`,
  to: 'user@example.com',
  subject: 'Subject',
  html: '<h1>Content</h1>'
})
```

🎯 LEAD CAPTURE FLOW

Lead Sources

1. **Calculators** (Primary)
   - AI Team Calculator
   - Automation Calculator
   - Hiring Risk Calculator
   - Profit Potential Calculator

2. **Contact Forms**
   - General contact form
   - Service-specific contact forms
   - Consultation booking form

3. **Exit Intent Popups**
   - Automation exit intent

   - Background checks exit intent
   - Business transformation exit intent

4. **Lead Magnets**
   - Automation guide
   - Background check checklist
   - ROI worksheets

Complete Flow

```
┌──────────────────────────────────────────────┐
│  1. USER INTERACTION                           │
│  - Fills calculator form                       │
│  - Submits contact form                        │
│  - Requests lead magnet                        │
└──────────────────────────────────────────────┘
                   │
                   │
                   ▼
┌──────────────────────────────────────────────┐
│  2. CLIENT-SIDE VALIDATION                     │
│  - Required fields check                       │
│  - Email format validation                     │
│  - Phone number validation                     │
└──────────────────────────────────────────────┘
                   │
                   │
                   ▼
┌──────────────────────────────────────────────┐
│  3. API REQUEST                                │
│  POST /api/calculator/recommendations          │
│  {                                             │
│    calculatorType: "ai-team",                  │
│    responses: { ... },                         │
│    leadData: { name, email, company }          │
│  }                                             │
└──────────────────────────────────────────────┘
                   │
                   │
                   ▼
┌──────────────────────────────────────────────┐
│  4. SERVER-SIDE PROCESSING                     │
│  a) Validate input data                        │
│  b) Calculate ROI projections                  │
│  c) Generate service recommendations           │
└──────────────────────────────────────────────┘
                   │
                   │
                   ▼
┌──────────────────────────────────────────────┐
│  5. DATABASE OPERATIONS                        │
│  a) Create/update Lead record                  │
│     - email (unique)                           │
│     - name, company, phone                     │
│     - source (calculator type)                 │
│     - calculatorData (JSON)                    │
│  b) Create CalculatorResponse record           │
│     - calculatorType                           │
│     - responses (form data)                    │
│     - recommendations                          │
│     - projectedSavings                         │
│     - leadId (foreign key)                     │
└──────────────────────────────────────────────┘
                   │
                   │
                   ▼
┌──────────────────────────────────────────────┐
│  6. EMAIL AUTOMATION                           │
│  a) Generate personalized email                │
│     - Include ROI results                      │
│     - Service recommendations                  │
│     - CTA to book consultation                 │
│  b) Send via Gmail API                         │
│  c) Log send status                            │
└──────────────────────────────────────────────┘
                   │
                   │
                   ▼
```

```
┌──────────────────────────────────────────────────────────────┐
│  7. DISCORD NOTIFICATION                                      │
│  a) Format embed message                                      │
│     - Title: "🎯 New Lead Captured"                           │
│     - Fields:                                                 │
│       * Name: {name}                                          │
│       * Email: {email}                                        │
│       * Company: {company}                                    │
│       * Source: {source}                                      │
│       * Projected Savings: ${amount}                          │
│  b) Send to Discord webhook                                   │
│  c) Log notification status                                   │
└──────────────────────────────────────────────────────────────┘
                         │
                         │
                         ▼
┌──────────────────────────────────────────────────────────────┐
│  8. CLIENT RESPONSE                                           │
│  {                                                            │
│    success: true,                                             │
│    recommendations: [ ... ],                                  │
│    projectedSavings: { ... }                                  │
│  }                                                            │
└──────────────────────────────────────────────────────────────┘
                         │
                         │
                         ▼
┌──────────────────────────────────────────────────────────────┐
│  9. UI UPDATE                                                 │
│  a) Show success modal                                        │
│  b) Display ROI results                                       │
│  c) Show service recommendations                              │
│  d) Provide CTA to book consultation                          │
└──────────────────────────────────────────────────────────────┘
```

Implementation Example

Calculator Component:

```
'use client'

import { useState } from 'react'
import { Button } from '@/components/ui/button'
import { Input } from '@/components/ui/input'

export function AITeamCalculator() {
  const [formData, setFormData] = useState({
    employees: '',
    avgSalary: '',
    name: '',
    email: '',
    company: ''
  })
  const [results, setResults] = useState(null)
  const [loading, setLoading] = useState(false)

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault()
    setLoading(true)

    try {
      const response = await fetch('/api/calculator/recommendations', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
          calculatorType: 'ai-team',
          responses: {
            employees: parseInt(formData.employees),
            avgSalary: parseFloat(formData.avgSalary)
          },
          leadData: {
            name: formData.name,
            email: formData.email,
            company: formData.company
          }
        })
      })

      const data = await response.json()

      if (data.success) {
        setResults(data)
        // Show results modal
      }
    } catch (error) {
      console.error('Calculator error:', error)
    } finally {
      setLoading(false)
    }
  }

  return (
    <form onSubmit={handleSubmit}>
      {/* Calculator inputs */}
      <Input
        type="number"
        placeholder="Number of employees"
        value={formData.employees}
        onChange={(e) => setFormData({ ...formData, employees: e.target.value })}
        required
      />
```

```
        <Input
          type="number"
          placeholder="Average salary"
          value={formData.avgSalary}
          onChange={(e) => setFormData({ ...formData, avgSalary: e.target.value })}
          required
        />

        {/* Lead capture fields */}
        <Input
          type="text"
          placeholder="Your name"
          value={formData.name}
          onChange={(e) => setFormData({ ...formData, name: e.target.value })}
          required
        />
        <Input
          type="email"
          placeholder="Email"
          value={formData.email}
          onChange={(e) => setFormData({ ...formData, email: e.target.value })}
          required
        />
        <Input
          type="text"
          placeholder="Company"
          value={formData.company}
          onChange={(e) => setFormData({ ...formData, company: e.target.value })}
        />

        <Button type="submit" disabled={loading}>
          {loading ? 'Calculating...' : 'See Your ROI'}
        </Button>
      </form>
    )
  }
```

API Route:

```
// app/api/calculator/recommendations/route.ts
import { NextRequest, NextResponse } from 'next/server'
import { db } from '@/lib/db'
import { sendEmail } from '@/lib/email'
import { sendDiscordNotification } from '@/lib/discord'
import { calculateROI } from '@/lib/calculator'

export async function POST(request: NextRequest) {
  const { calculatorType, responses, leadData } = await request.json()

  // Calculate ROI
  const projectedSavings = calculateROI(calculatorType, responses)

  // Generate recommendations
  const recommendations = generateRecommendations(calculatorType, responses)

  // Create/update lead
  const lead = await db.lead.upsert({
    where: { email: leadData.email },
    update: {
      name: leadData.name,
      company: leadData.company,
      calculatorData: responses,
      source: `${calculatorType}-calculator`,
      updatedAt: new Date()
    },
    create: {
      email: leadData.email,
      name: leadData.name,
      company: leadData.company,
      calculatorData: responses,
      source: `${calculatorType}-calculator`
    }
  })

  // Create calculator response
  await db.calculatorResponse.create({
    data: {
      calculatorType,
      responses,
      recommendations,
      projectedSavings,
      leadId: lead.id
    }
  })

  // Send confirmation email
  await sendEmail({
    to: lead.email,
    subject: 'Your ROI Analysis Results',
    html: generateEmailTemplate(lead, projectedSavings, recommendations),
    text: generateEmailText(lead, projectedSavings, recommendations)
  })

  // Send Discord notification
  await sendDiscordNotification({
    title: '🎯 New Lead Captured',
    description: `${lead.name} from ${lead.company || 'Unknown Company'}`,
    fields: [
      { name: 'Email', value: lead.email },
      { name: 'Source', value: lead.source },
      { name: 'Projected Savings', value: `$${projectedSavings.annual:,}/year` }
    ],
    color: 0x7c3aed // Purple
  })

  return NextResponse.json({
    success: true,
    recommendations,
    projectedSavings
  })
}
```

Lead Scoring (Future Implementation)

Scoring Criteria:
- Calculator completion: +10 points
- Email verification: +5 points
- Consultation booking: +20 points
- High ROI projections (>$100K/year): +15 points
- Company size (enterprise): +10 points
- Multiple interactions: +5 points each

Priority Levels:
- **Hot Lead (50+ points):** Immediate follow-up
- **Warm Lead (30-49 points):** Follow-up within 24 hours
- **Cold Lead (<30 points):** Email nurture campaign

📊 CALCULATOR SYSTEMS

Calculator Overview

Total Calculators: 4

1. **AI Team Calculator** - Employee replacement ROI
2. **Automation Calculator** - Process automation savings
3. **Hiring Risk Calculator** - Bad hire cost analysis
4. **Profit Potential Calculator** - Business transformation ROI

1. AI Team Calculator

File: `components/calculator/ai-team-calculator.tsx`

Purpose: Calculate savings from replacing employees with AI

Inputs:
- **Number of employees:** 1-100
- **Average salary:** $20,000-$200,000
- **Tasks performed:** Multi-select
- Data entry
- Customer support
- Content creation
- Research
- Scheduling

- Reporting
- **Hours per week:** 1-168

Calculation Logic:

```
function calculateAITeamROI(inputs) {
  const annualSalary = inputs.avgSalary * inputs.employees
  const aiCost = inputs.employees * 5000 // $5K per AI employee/year
  const annualSavings = annualSalary - aiCost
  const fiveYearSavings = annualSavings * 5
  const roi = ((annualSavings / aiCost) * 100).toFixed(0)

  return {
    monthly: annualSavings / 12,
    annual: annualSavings,
    fiveYear: fiveYearSavings,
    roi: roi + '%',
    paybackPeriod: (aiCost / annualSavings * 12).toFixed(1) + ' months'
  }
}
```

Recommendations:

```
function generateAITeamRecommendations(inputs) {
  const recommendations = []

  if (inputs.tasks.includes('customer-support')) {
    recommendations.push({
      service: 'AI Workforce - Customer Support',
      reason: 'High volume of customer support tasks',
      estimatedROI: '400%',
      timeFrame: '3-6 months'
    })
  }

  if (inputs.tasks.includes('data-entry')) {
    recommendations.push({
      service: 'Intelligent Automation',
      reason: 'Repetitive data entry can be fully automated',
      estimatedROI: '600%',
      timeFrame: '1-3 months'
    })
  }

  // ... more logic

  return recommendations
}
```

2. Automation Calculator

File: `components/calculator/automation-calculator.tsx`

Purpose: Calculate time and cost savings from process automation

Inputs:
- **Process hours per week:** 1-168
- **Number of employees:** 1-100

- **Average hourly rate:** $10-$150
- **Process type:** Dropdown
- Data processing
- Report generation
- Email management
- Document processing
- Invoice processing
- Customer onboarding

Calculation Logic:

```
function calculateAutomationROI(inputs) {
  const weeklyHours = inputs.processHours * inputs.employees
  const annualHours = weeklyHours * 52
  const hourlyCost = inputs.avgHourlyRate
  const annualCost = annualHours * hourlyCost
  const automationCost = 10000 // $10K setup + $200/mo
  const annualSavings = annualCost * 0.7 // 70% automation efficiency
  const fiveYearSavings = (annualSavings * 5) - automationCost

  return {
    monthly: annualSavings / 12,
    annual: annualSavings,
    fiveYear: fiveYearSavings,
    timeRecovered: weeklyHours + ' hours/week',
    roi: ((annualSavings / automationCost) * 100).toFixed(0) + '%'
  }
}
```

3. Hiring Risk Calculator

File: `components/calculator/hiring-risk-calculator.tsx`

Purpose: Calculate cost of bad hires and value of background checks

Inputs:
- **Position level:** Dropdown
- Entry level
- Mid-level
- Senior level
- Executive
- **Annual salary:** $30,000-$500,000
- **Hiring process cost:** $5,000-$50,000
- **Time to productivity:** 1-12 months

Calculation Logic:

```
function calculateHiringRisk(inputs) {
  const salary = inputs.salary
  const hiringCost = inputs.hiringProcessCost
  const timeToProductivity = inputs.timeToProductivity
  const lostProductivity = (salary / 12) * timeToProductivity * 0.5
  const replacementCost = hiringCost
  const reputationDamage = salary * 0.2 // 20% of salary

  const totalBadHireCost =
    salary +
    hiringCost +
    lostProductivity +
    replacementCost +
    reputationDamage

  const backgroundCheckCost = 5000
  const roi = ((totalBadHireCost / backgroundCheckCost) * 100).toFixed(0)

  return {
    totalCost: totalBadHireCost,
    breakdownbreak: {
      salary: salary,
      hiringCost: hiringCost,
      lostProductivity: lostProductivity,
      replacementCost: replacementCost,
      reputationDamage: reputationDamage
    },
    backgroundCheckCost: backgroundCheckCost,
    roi: roi + '%',
    riskLevel: totalBadHireCost > 200000 ? 'High' : 'Medium'
  }
}
```

Recommendations:

```
function generateHiringRecommendations(inputs) {
  if (inputs.positionLevel === 'Executive') {
    return [{
      service: 'Beyond Background Checks - Executive Vetting',
      reason: 'Executive hires require deep investigations',
      recommendedPackage: 'Comprehensive Executive Package',
      cost: '$15,000',
      timeline: '7-14 days'
    }]
  }

  if (inputs.salary > 100000) {
    return [{
      service: 'Beyond Background Checks - Professional Package',
      reason: 'High-salary positions warrant thorough vetting',
      recommendedPackage: 'Professional Package',
      cost: '$7,500',
      timeline: '5-7 days'
    }]
  }

  // Default
  return [{
    service: 'Beyond Background Checks - Standard Package',
    reason: 'Prevent costly bad hires',
    recommendedPackage: 'Standard Package',
    cost: '$5,000',
    timeline: '3-5 days'
  }]
}
```

4. Profit Potential Calculator

File: `components/calculator/profit-potential-calculator.tsx`

Purpose: Calculate business transformation ROI and profit increase

Inputs:
- **Current annual revenue:** $100K-$50M
- **Current profit margin:** 0-50%
- **Industry:** Dropdown
- Technology
- Healthcare
- Finance
- Retail
- Manufacturing
- Professional Services
- **Growth goals:** 1-5x
- **Transformation areas:** Multi-select
- Operations optimization
- Customer experience
- Data analytics
- AI integration
- Process automation

Calculation Logic:

```javascript
function calculateProfitPotential(inputs) {
  const currentRevenue = inputs.revenue
  const currentProfit = currentRevenue * (inputs.profitMargin / 100)
  const transformationCost = currentRevenue * 0.05 // 5% of revenue

  // Industry-specific multipliers
  const industryMultipliers = {
    'Technology': 1.4,
    'Healthcare': 1.2,
    'Finance': 1.3,
    'Retail': 1.15,
    'Manufacturing': 1.25,
    'Professional Services': 1.35
  }

  const multiplier = industryMultipliers[inputs.industry] || 1.2
  const potentialRevenue = currentRevenue * inputs.growthGoals * multiplier
  const potentialProfit = potentialRevenue * ((inputs.profitMargin + 10) / 100)
  const profitIncrease = potentialProfit - currentProfit
  const roi = ((profitIncrease / transformationCost) * 100).toFixed(0)

  return {
    currentProfit: currentProfit,
    potentialProfit: potentialProfit,
    profitIncrease: profitIncrease,
    transformationCost: transformationCost,
    roi: roi + '%',
    timeline: '12-24 months'
  }
}
```

Recommendations:

```
function generateTransformationRecommendations(inputs) {
  const recommendations = []

  if (inputs.transformationAreas.includes('process-automation')) {
    recommendations.push({
      service: 'Intelligent Automation',
      reason: 'Automate manual processes for efficiency gains',
      estimatedImpact: '30-40% cost reduction',
      timeline: '6-12 months'
    })
  }

  if (inputs.transformationAreas.includes('ai-integration')) {
    recommendations.push({
      service: 'AI Workforce',
      reason: 'Deploy AI employees for 24/7 operations',
      estimatedImpact: '50-70% productivity increase',
      timeline: '3-6 months'
    })
  }

  if (inputs.transformationAreas.includes('data-analytics')) {
    recommendations.push({
      service: 'Business Transformation',
      reason: 'Data-driven decision making improves margins',
      estimatedImpact: '15-25% revenue growth',
      timeline: '12-18 months'
    })
  }

  return recommendations
}
```

Calculator Best Practices

1. **Clear Value Proposition**
   - Show immediate value (ROI projections)
   - Use realistic, conservative estimates
   - Provide breakdowns and explanations

2. **Lead Capture Integration**
   - Require email to see results
   - Make form fields optional where possible
   - Provide immediate value (don’t gate too heavily)

3. **Progressive Disclosure**
   - Start with simple inputs
   - Show advanced options on demand
   - Don’t overwhelm with too many fields

4. **Visual Feedback**
   - Show loading states
   - Animate number transitions
   - Use progress indicators for multi-step

5. **Mobile Optimization**
   - Responsive design

   - Touch-friendly inputs
   - Simplified layout on mobile

6. **Conversion Optimization**
   - Clear CTA after results
   - Show social proof (testimonials)
   - Create urgency (limited consultation slots)

[Due to length constraints, I'll create a second file for the remaining sections]

This document continues in COMPLETE_DEVELOPER_HANDOVER_PART2.md