🚀 COMPLETE DEVELOPER HANDOVER DOCUMENT (PART 2)

QuantumLeap AI Website - Full Technical Documentation

Continued from COMPLETE_DEVELOPER_HANDOVER.md

📝 BLOG SYSTEM

Blog Architecture

Content Source: Markdown files OR Database (flexible)

Current Implementation: Static markdown files in `/public/blog/` (future: database)

URL Structure: `/blog/[slug]`

Features:
- Dynamic routing
- Markdown rendering
- Syntax highlighting (for code)
- SEO optimization
- Social sharing
- Related posts
- Author profiles
- Category/tag filtering

Blog Post Structure

Markdown Frontmatter:

```
---
title: "The Hacker Who Passed Every Background Check"
slug: "hacker-passed-background-check"
description: "How a $5K investigation could have prevented a $846K disaster"
author: "Paras Khurana"
date: "2025-10-15"
category: "Cyber Security"
tags: ["background-checks", "cyber-security", "case-study"]
featured_image: "/blog/hacker-case-study.jpg"
reading_time: "8 min read"
---

## Introduction

[Blog content in Markdown format]

### Key Takeaways

- Point 1
- Point 2
- Point 3

## Conclusion

[Conclusion content]
```

Rendering:

```tsx
// app/blog/[slug]/page.tsx
import { getPostBySlug, getAllPosts } from '@/lib/blog'
import ReactMarkdown from 'react-markdown'

export async function generateStaticParams() {
  const posts = await getAllPosts()
  return posts.map(post => ({ slug: post.slug }))
}

export default async function BlogPost({ params }: { params: { slug: string } }) {
  const post = await getPostBySlug(params.slug)

  return (
    <article>
      <header>
        <h1>{post.title}</h1>
        <p>{post.description}</p>
        <div>
          <span>{post.author}</span>
          <span>{post.date}</span>
          <span>{post.reading_time}</span>
        </div>
      </header>

      <div>
        <ReactMarkdown>{post.content}</ReactMarkdown>
      </div>

      <footer>
        {/* Related posts, share buttons, CTA */}
      </footer>
    </article>
  )
}
```

Blog API (Admin)

File: `app/api/admin/blog-api-keys/route.ts`

Purpose: Manage API keys for blog content creation/editing

Endpoints:

GET /api/admin/blog-api-keys

Purpose: List all API keys

Response:

```
{
  "keys": [
    {
      "id": "uuid",
      "keyName": "Content Writer Key",
      "apiKey": "qla_xxx",
      "isActive": true,
      "usageCount": 42,
      "createdAt": "2025-11-01",
      "lastUsedAt": "2025-11-10"
    }
  ]
}
```

POST /api/admin/blog-api-keys

Purpose: Create new API key

Request:

```
{
  "keyName": "New Content Writer"
}
```

Response:

```
{
  "success": true,
  "apiKey": "qla_newkeyhere",
  "keyName": "New Content Writer"
}
```

DELETE /api/admin/blog-api-keys

Purpose: Revoke API key

Request:

```
{
  "apiKey": "qla_keytorevoke"
}
```

Blog Creation API

File: `app/api/blog/post/route.ts`

Purpose: Create/update blog posts via API

POST /api/blog/post

Authentication: API Key in header

Request:

```
{
  "title": "New Blog Post",
  "slug": "new-blog-post",
  "description": "Short description",
  "content": "# Markdown content here",
  "author": "Paras Khurana",
  "category": "AI",
  "tags": ["ai", "automation"],
  "featured_image": "/blog/image.jpg"
}
```

Response:

```
{
  "success": true,
  "postId": "uuid",
  "slug": "new-blog-post",
  "url": "https://quantumleapai.abacusai.app/blog/new-blog-post"
}
```

SEO Optimization

Meta Tags:

```
export const metadata = {
  title: post.title,
  description: post.description,
  openGraph: {
    title: post.title,
    description: post.description,
    images: [post.featured_image],
    type: 'article',
    publishedTime: post.date,
    authors: [post.author]
  },
  twitter: {
    card: 'summary_large_image',
    title: post.title,
    description: post.description,
    images: [post.featured_image]
  }
}
```

Schema Markup:

```
<script type="application/ld+json">
  {JSON.stringify({
    "@context": "https://schema.org",
    "@type": "BlogPosting",
    "headline": post.title,
    "description": post.description,
    "image": post.featured_image,
    "datePublished": post.date,
    "author": {
      "@type": "Person",
      "name": post.author
    },
    "publisher": {
      "@type": "Organization",
      "name": "QuantumLeap AI",
      "logo": {
        "@type": "ImageObject",
        "url": "https://quantumleapai.abacusai.app/logo.png"
      }
    }
  })}
</script>
```

Content Strategy

Content Pillars:
1. **AI & Automation** - AI workforce, automation case studies
2. **Cybersecurity** - Threat intelligence, breach prevention
3. **Hiring & HR** - Background checks, hiring risks
4. **Business Growth** - Transformation stories, ROI analysis

Post Types:
- **Case Studies** - Real client stories with metrics
- **How-To Guides** - Step-by-step tutorials
- **Industry Analysis** - Market trends and insights
- **Thought Leadership** - Founder's perspective

Publishing Schedule:
- **Frequency:** 2-3 posts per week
- **Best Days:** Tuesday, Wednesday, Thursday
- **Best Times:** 10 AM - 2 PM EST

🎛️ ADMIN DASHBOARD

Dashboard Overview

URL: `/admin`

Authentication: Required (JWT)

Layout: `app/admin/layout.tsx`

Features:
- Real-time analytics
- Lead management

- Consultation management
- Calendar integration
- Blog management
- CRM functionality

Analytics Dashboard

File: `app/admin/page.tsx`

Metrics Displayed:

Summary Cards

```
<div className="grid grid-cols-4 gap-6">
  <Card>
    <h3>Total Leads</h3>
    <p className="text-4xl">{totalLeads}</p>
    <p className="text-sm text-muted">+{newLeadsToday} today</p>
  </Card>

  <Card>
    <h3>Consultations Booked</h3>
    <p className="text-4xl">{consultationsBooked}</p>
    <p className="text-sm text-muted">+{newBookingsToday} today</p>
  </Card>

  <Card>
    <h3>Calculator Completions</h3>
    <p className="text-4xl">{calculatorCompletions}</p>
    <p className="text-sm text-muted">{completionRate}% completion rate</p>
  </Card>

  <Card>
    <h3>Conversion Rate</h3>
    <p className="text-4xl">{conversionRate}%</p>
    <p className="text-sm text-muted">Lead to consultation</p>
  </Card>
</div>
```

Lead Sources Chart

```
<Card>
  <h3>Leads by Source</h3>
  <PieChart
    data={[
      { name: 'AI Team Calculator', value: 50 },
      { name: 'Automation Calculator', value: 40 },
      { name: 'Contact Form', value: 30 },
      { name: 'Lead Magnet', value: 30 }
    ]}
  />
</Card>
```

Recent Activity

```
<Card>
  <h3>Recent Activity</h3>
  <ul>
    {recentActivity.map(activity => (
      <li key={activity.id}>
        <span>{activity.type}</span>
        <span>{activity.description}</span>
        <span>{activity.timestamp}</span>
      </li>
    ))}
  </ul>
</Card>
```

Lead Management

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

Features:
- Sortable table
- Filter by source/status/date
- Search by name/email/company
- Bulk actions (export, update status)
- Lead details modal
- Quick actions (email, call, add note)

Table Columns:
- Checkbox (select)
- Name
- Email
- Company
- Source
- Confirmed (✓/✗)
- Created At
- Actions

Filters:

```
<div className="flex gap-4">
  <Select>
    <option value="all">All Sources</option>
    <option value="ai-team-calculator">AI Team Calculator</option>
    <option value="automation-calculator">Automation Calculator</option>
    <option value="contact-form">Contact Form</option>
  </Select>

  <Select>
    <option value="all">All Status</option>
    <option value="confirmed">Confirmed</option>
    <option value="unconfirmed">Unconfirmed</option>
  </Select>

  <Input
    type="search"
    placeholder="Search by name, email, or company"
  />
</div>
```

Bulk Actions:

```
<Button onClick={exportToCSV}>
  Export to CSV
</Button>

<Button onClick={updateSelectedStatus}>
  Mark as Contacted
</Button>
```

Consultation Management

File: `app/admin/consultations/route.ts`

Features:
- Calendar view
- List view
- Filter by status/service/date
- Booking details
- Update status (pending → confirmed → completed)
- Reschedule
- Cancel

Booking Card:

```
<Card>
  <div className="flex justify-between">
    <div>
      <h3>{booking.name}</h3>
      <p>{booking.email}</p>
      <p>{booking.company}</p>
    </div>
    <Badge>{booking.status}</Badge>
  </div>

  <div>
    <p><strong>Service:</strong> {booking.serviceInterest}</p>
    <p><strong>Date:</strong> {booking.preferredDate}</p>
    <p><strong>Time:</strong> {booking.preferredTime}</p>
  </div>

  {booking.message && (
    <div>
      <p><strong>Message:</strong></p>
      <p>{booking.message}</p>
    </div>
  )}

  <div className="flex gap-2">
    <Button>Confirm</Button>
    <Button variant="outline">Reschedule</Button>
    <Button variant="destructive">Cancel</Button>
  </div>
</Card>
```

Calendar Integration

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

Features:
- Google Calendar sync
- Day/week/month views
- Drag-and-drop events
- Event creation
- Event editing
- Attendee management

Implementation:

```tsx
'use client'

import { Calendar } from '@/components/ui/calendar'
import { useState, useEffect } from 'react'

export default function AdminCalendar() {
  const [events, setEvents] = useState([])
  const [view, setView] = useState('month')

  useEffect(() => {
    fetch('/api/admin/calendar')
      .then(res => res.json())
      .then(data => setEvents(data.events))
  }, [])

  const handleEventCreate = async (event) => {
    const response = await fetch('/api/admin/calendar', {
      method: 'POST',
      body: JSON.stringify(event)
    })

    if (response.ok) {
      // Refresh events
    }
  }

  return (
    <div>
      <div className="flex justify-between mb-4">
        <h1>Calendar</h1>
        <div className="flex gap-2">
          <Button onClick={() => setView('day')}>Day</Button>
          <Button onClick={() => setView('week')}>Week</Button>
          <Button onClick={() => setView('month')}>Month</Button>
        </div>
      </div>

      <Calendar
        events={events}
        view={view}
        onEventClick={handleEventClick}
        onEventCreate={handleEventCreate}
        onEventUpdate={handleEventUpdate}
      />
    </div>
  )
}
```

CRM Features

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

Pipeline Stages:

1. **New Lead** - Just captured
2. **Qualified** - Meets criteria
3. **Contacted** - Initial outreach done
4. **Consultation Booked** - Meeting scheduled
5. **Proposal Sent** - Quote provided
6. **Negotiation** - Discussing terms

7. **Won** - Deal closed
8. **Lost** - Deal lost

Kanban View:

```
<div className="grid grid-cols-7 gap-4">
  {stages.map(stage => (
    <div key={stage.id} className="bg-card p-4 rounded-lg">
      <h3>{stage.name}</h3>
      <p className="text-sm text-muted">{stage.count} leads</p>

      <div className="space-y-2 mt-4">
        {stage.leads.map(lead => (
          <Card
            key={lead.id}
            draggable
            onDragStart={() => handleDragStart(lead)}
            onDrop={() => handleDrop(stage)}
          >
            <h4>{lead.name}</h4>
            <p className="text-sm">{lead.company}</p>
            <p className="text-xs text-muted">{lead.source}</p>
            <Badge>{lead.value}</Badge>
          </Card>
        ))}
      </div>
    </div>
  ))}
</div>
```

🚢 BUILD AND DEPLOYMENT

Build Process

Production Build:

```
cd nextjs_space
npm run build
```

What Happens:
1. TypeScript compilation
2. Code optimization
3. Bundle splitting
4. Image optimization
5. Static page generation
6. Route pre-rendering

Build Output:

```
.next/
├── cache/                  # Build cache
├── server/                 # Server-side code
│   ├── app/               # App router pages
│   ├── chunks/            # Shared code chunks
│   └── pages/             # API routes
├── static/                # Static assets
│   ├── chunks/            # JS chunks
│   ├── css/               # Stylesheets
│   └── media/             # Images, fonts
└── build-manifest.json    # Build metadata
```

Build Stats:

```
Route (app)                                 Size     First Load JS
┌ ○ /                                       5.2 kB         92.1 kB
├ ○ /about-us                               3.8 kB         89.7 kB
├ ○ /ai-workforce                           4.5 kB         90.4 kB
├ ○ /background-checks                      4.1 kB         90.0 kB
├ ○ /blog                                   2.9 kB         88.8 kB
├ ƒ /blog/[slug]                            3.5 kB         89.4 kB
├ ○ /business-transformation                4.2 kB         90.1 kB
├ ○ /consultation                           3.6 kB         89.5 kB
├ ○ /cyber-intelligence                     4.0 kB         89.9 kB
├ ○ /intelligent-automation                 4.3 kB         90.2 kB
├ ○ /smb                                    5.8 kB         91.7 kB
└ ○ /[...other routes]

○ (Static)  prerendered as static content
ƒ (Dynamic) server-rendered on demand
```

Deployment Platforms

1. Abacus.AI (Current)

Live URL: https://quantumleapai.abacusai.app

Deployment Method: Automated via Abacus.AI platform

Configuration:
- Auto-deploy on push to `semi-final` branch
- Environment variables configured in dashboard
- PostgreSQL database (Supabase) connected
- CDN enabled for static assets

Deployment Steps:
1. Commit changes to `semi-final` branch
2. Push to GitHub
3. Abacus.AI auto-detects changes
4. Runs build process
5. Deploys to production
6. Updates DNS

2. Vercel (Alternative)

Setup:

```
# Install Vercel CLI
npm i -g vercel

# Login
vercel login

# Deploy
vercel --prod
```

Configuration (`vercel.json`):

```
{
  "buildCommand": "cd nextjs_space && npm run build",
  "outputDirectory": "nextjs_space/.next",
  "framework": "nextjs",
  "regions": ["iad1"],
  "env": {
    "DATABASE_URL": "@database-url",
    "GMAIL_CLIENT_EMAIL": "@gmail-client-email",
    "GMAIL_PRIVATE_KEY": "@gmail-private-key"
  }
}
```

Environment Variables:
- Set via Vercel dashboard or CLI
- Use secrets for sensitive data
- Separate production/preview environments

3. Netlify (Alternative)

Setup:

```
# Install Netlify CLI
npm i -g netlify-cli

# Login
netlify login

# Deploy
netlify deploy --prod
```

Configuration (`netlify.toml`):

```
[build]
  command = "cd nextjs_space && npm run build"
  publish = "nextjs_space/.next"

[build.environment]
  NODE_VERSION = "18"

[[redirects]]
  from = "/*"
  to = "/index.html"
  status = 200
```

4. Custom Server (VPS/Dedicated)

Requirements:

- Node.js 18+
- nginx (reverse proxy)
- PM2 (process manager)
- SSL certificate (Let’s Encrypt)

Setup:

```
# 1. Install dependencies
sudo apt update
sudo apt install nodejs npm nginx certbot python3-certbot-nginx

# 2. Clone repository
git clone https://github.com/AICyberSecurity911/WBsite.git
cd WBsite
git checkout semi-final
cd nextjs_space

# 3. Install packages
npm install --legacy-peer-deps

# 4. Build
npm run build

# 5. Install PM2
npm install -g pm2

# 6. Start app
pm2 start npm --name "quantumleap" -- start

# 7. Configure nginx
sudo nano /etc/nginx/sites-available/quantumleap

# Add:
server {
    listen 80;
    server_name quantumleap.io www.quantumleap.io;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}

# 8. Enable site
sudo ln -s /etc/nginx/sites-available/quantumleap /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl reload nginx

# 9. Get SSL certificate
sudo certbot --nginx -d quantumleap.io -d www.quantumleap.io

# 10. Save PM2 startup script
pm2 save
pm2 startup
```

Environment Management

Development (.env.local):

```
NODE_ENV=development
NEXT_PUBLIC_BASE_URL=http://localhost:3000
DATABASE_URL=postgresql://... (dev database)
```

Production (.env.production):

```
NODE_ENV=production
NEXT_PUBLIC_BASE_URL=https://quantumleapai.abacusai.app
DATABASE_URL=postgresql://... (prod database)
```

Best Practices:
- Never commit `.env` files
- Use `.env.example` as template
- Store secrets in platform dashboards
- Rotate secrets regularly
- Use different databases for dev/prod
- Test env vars before deployment

CI/CD Pipeline (GitHub Actions)

File: `.github/workflows/deploy.yml`

```yaml
name: Deploy to Production

on:
  push:
    branches: [semi-final]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'

      - name: Install dependencies
        run: |
          cd nextjs_space
          npm install --legacy-peer-deps

      - name: Run TypeScript check
        run: |
          cd nextjs_space
          npx tsc --noEmit

      - name: Run ESLint
        run: |
          cd nextjs_space
          npm run lint

      - name: Build
        run: |
          cd nextjs_space
          npm run build

  deploy:
    needs: test
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/semi-final'
    steps:
      - uses: actions/checkout@v3

      - name: Deploy to Abacus.AI
        env:
          DEPLOY_KEY: ${{ secrets.DEPLOY_KEY }}
        run: |
          # Deployment script here
```

Pre-Deployment Checklist

Code:

- [ ] All TypeScript errors resolved
- [ ] ESLint warnings addressed
- [ ] Build completes successfully
- [ ] No console errors in dev tools
- [ ] All API routes tested

Environment:
- [ ] Environment variables set
- [ ] Database connection verified
- [ ] API keys configured
- [ ] Secrets rotated

Content:
- [ ] All images optimized
- [ ] Meta tags updated
- [ ] Sitemap generated
- [ ] robots.txt configured

Testing:
- [ ] Cross-browser tested (Chrome, Firefox, Safari, Edge)
- [ ] Mobile responsive
- [ ] Calculators functional
- [ ] Forms submitting correctly
- [ ] Emails sending

Performance:
- [ ] Lighthouse score > 90
- [ ] Images lazy-loaded
- [ ] Bundle size optimized
- [ ] Caching configured

SEO:
- [ ] Meta descriptions unique
- [ ] H1 tags present on all pages
- [ ] Schema markup validated
- [ ] Canonical URLs set
- [ ] 404 page exists

Security:
- [ ] HTTPS enabled
- [ ] Environment variables secured
- [ ] API keys not exposed in client
- [ ] CORS configured
- [ ] Rate limiting enabled

Post-Deployment Verification

```
# 1. Check if site is live
curl -I https://quantumleapai.abacusai.app

# 2. Test API endpoints
curl https://quantumleapai.abacusai.app/api/health

# 3. Check database connection
# (via admin dashboard or Prisma Studio)

# 4. Verify email sending
# (submit test form)

# 5. Check Discord notifications
# (submit test lead)

# 6. Test calculators
# (complete each calculator)

# 7. Monitor error logs
# (check hosting platform dashboard)

# 8. Run Lighthouse audit
# (in Chrome DevTools)
```

✅ TESTING AND QA

Testing Strategy

Types of Testing:

1. **Unit Testing** - Individual functions (future)
2. **Integration Testing** - API routes, database operations (future)
3. **E2E Testing** - User flows (future)
4. **Manual Testing** - Current approach

Manual Testing Checklist

Homepage Tests

- [ ] Hero video autoplays and transitions
- [ ] Both CTAs link to `/smb`
- [ ] Trust bar logos display
- [ ] Exit intent popup appears on mouse leave
- [ ] Breach checker validates email format
- [ ] Breach checker shows results
- [ ] Calculator section scrolls smoothly
- [ ] Testimonials carousel rotates
- [ ] All navigation links work
- [ ] Dark/light mode toggle works
- [ ] Mobile menu opens/closes
- [ ] Footer links work
- [ ] Newsletter signup works

Service Page Tests (Repeat for all 5 services)

- [ ] Hero section displays correctly
- [ ] Calculator loads and functions
- [ ] Calculator validates inputs
- [ ] Calculator shows results
- [ ] Lead capture form submits
- [ ] Confirmation email received
- [ ] Discord notification sent
- [ ] Guarantee section displays
- [ ] Testimonials load
- [ ] FAQ accordion expands/collapses
- [ ] CTA button links correctly
- [ ] Exit intent popup appears
- [ ] SEO metadata present

Blog Tests

- [ ] Blog listing loads
- [ ] Blog post opens
- [ ] Markdown renders correctly
- [ ] Code blocks syntax highlighted
- [ ] Images load
- [ ] Related posts show
- [ ] Social share buttons work
- [ ] CTA section displays

Admin Dashboard Tests

- [ ] Login page loads
- [ ] Authentication works
- [ ] Dashboard displays metrics
- [ ] Leads table loads
- [ ] Lead filtering works
- [ ] Lead search works
- [ ] Lead export to CSV works
- [ ] Consultation list loads
- [ ] Calendar loads events
- [ ] Event creation works
- [ ] CRM pipeline displays
- [ ] Logout works

Form Tests

- [ ] Required field validation
- [ ] Email format validation
- [ ] Phone format validation
- [ ] Success message displays
- [ ] Error handling works
- [ ] Loading states show

- [ ] Submission stores in database
- [ ] Confirmation email sent

Cross-Browser Tests

- [ ] Chrome (latest)
- [ ] Firefox (latest)
- [ ] Safari (latest)
- [ ] Edge (latest)
- [ ] Mobile Chrome
- [ ] Mobile Safari

Responsive Tests

- [ ] Mobile (320px-480px)
- [ ] Tablet (768px-1024px)
- [ ] Desktop (1280px+)
- [ ] Ultra-wide (1920px+)

Automated Testing (Future Implementation)

Testing Tools:
- **Jest** - Unit testing
- **React Testing Library** - Component testing
- **Playwright** - E2E testing
- **Cypress** - E2E testing (alternative)

Example Unit Test:

```
// lib/calculator.test.ts
import { calculateAITeamROI } from './calculator'

describe('calculateAITeamROI', () => {
  it('should calculate correct annual savings', () => {
    const result = calculateAITeamROI({
      employees: 5,
      avgSalary: 50000
    })

    expect(result.annual).toBe(225000) // (50000 * 5) - (5 * 5000)
  })

  it('should calculate correct ROI percentage', () => {
    const result = calculateAITeamROI({
      employees: 5,
      avgSalary: 50000
    })

    expect(result.roi).toBe('900%') // (225000 / 25000) * 100
  })
})
```

Example E2E Test:

```
// tests/e2e/calculator.spec.ts
import { test, expect } from '@playwright/test'

test('AI Team Calculator flow', async ({ page }) => {
  await page.goto('https://quantumleapai.abacusai.app/ai-workforce')

  // Fill calculator
  await page.fill('input[name="employees"]', '5')
  await page.fill('input[name="avgSalary"]', '50000')

  // Fill lead capture
  await page.fill('input[name="name"]', 'Test User')
  await page.fill('input[name="email"]', 'test@example.com')
  await page.fill('input[name="company"]', 'Test Co')

  // Submit
  await page.click('button[type="submit"]')

  // Wait for results
  await expect(page.locator('.results-modal')).toBeVisible()

  // Check if annual savings displayed
  await expect(page.locator('.annual-savings')).toContainText('$225,000')
})
```

Performance Testing

Tools:
- **Lighthouse** - Overall performance audit
- **WebPageTest** - Detailed performance metrics
- **GTmetrix** - Speed and optimization analysis

Target Metrics:
- **Performance:** 90+
- **Accessibility:** 95+
- **Best Practices:** 95+
- **SEO:** 100
- **First Contentful Paint:** < 1.5s
- **Largest Contentful Paint:** < 2.5s
- **Time to Interactive:** < 3.5s
- **Cumulative Layout Shift:** < 0.1
- **Total Blocking Time:** < 300ms

Running Lighthouse:

```
# In Chrome DevTools
1. Open DevTools (F12)
2. Go to "Lighthouse" tab
3. Select "Desktop" or "Mobile"
4. Click "Generate report"

# Via CLI
npm install -g lighthouse
lighthouse https://quantumleapai.abacusai.app --view
```

⚡ PERFORMANCE OPTIMIZATION

Image Optimization

Current Implementation: Next.js Image component

Best Practices:

```
import Image from 'next/image'

// ✅ Correct
<Image
  src="/hero-bg.jpg"
  alt="Hero background"
  width={1920}
  height={1080}
  priority // For above-the-fold images
  quality={85}
/>

// ✅ Correct (responsive)
<div className="relative aspect-video">
  <Image
    src="/image.jpg"
    alt="Description"
    fill
    className="object-cover"
  />
</div>

// ❌ Wrong (missing dimensions)
<Image src="/image.jpg" alt="Description" />
```

Image Formats:
- Use WebP for photos (smaller size)
- Use PNG for graphics with transparency
- Use SVG for logos and icons
- Use AVIF for cutting-edge browsers (future)

Image Sizes:
- Hero images: 1920x1080 (max 200KB)
- Card images: 600x400 (max 50KB)
- Thumbnails: 300x200 (max 20KB)
- Icons: 64x64 (SVG preferred)

Image Loading:
- `priority` for above-the-fold images
- `loading="lazy"` for below-the-fold
- `placeholder="blur"` for better UX

Code Splitting

Automatic: Next.js automatically splits code by route

Manual (for large components):

```
import dynamic from 'next/dynamic'

// Lazy load heavy component
const HeavyCalculator = dynamic(
  () => import('@/components/calculator/heavy-calculator'),
  {
    loading: () => <div>Loading calculator...</div>,
    ssr: false // Disable SSR if not needed
  }
)
```

Bundle Analysis

Install:

```
npm install @next/bundle-analyzer
```

Configure (`next.config.js`):

```
const withBundleAnalyzer = require('@next/bundle-analyzer')({
  enabled: process.env.ANALYZE === 'true'
})

module.exports = withBundleAnalyzer({
  // Next.js config
})
```

Run:

```
ANALYZE=true npm run build
```

Optimization Tips:
- Remove unused dependencies
- Use tree-shakable imports (`import { Button } from '@/components/ui/button'`)
- Lazy load heavy components
- Use dynamic imports for route-based code splitting

Caching Strategy

Static Assets:

```
// next.config.js
module.exports = {
  async headers() {
    return [
      {
        source: '/:all*(svg|jpg|png|webp|avif|woff|woff2)',
        headers: [
          {
            key: 'Cache-Control',
            value: 'public, max-age=31536000, immutable'
          }
        ]
      }
    ]
  }
}
```

API Routes:

```
export async function GET(request: NextRequest) {
  const data = await fetchData()

  return NextResponse.json(data, {
    headers: {
      'Cache-Control': 'public, s-maxage=3600, stale-while-revalidate=86400'
    }
  })
}
```

Database Queries:

```
import { cache } from 'react'

// Cache for duration of request
export const getLeads = cache(async () => {
  return await db.lead.findMany()
})
```

Font Optimization

Current Implementation: next/font

```tsx
// app/layout.tsx
import { Inter, Manrope } from 'next/font/google'

const inter = Inter({
  subsets: ['latin'],
  variable: '--font-inter',
  display: 'swap'
})

const manrope = Manrope({
  subsets: ['latin'],
  variable: '--font-manrope',
  display: 'swap'
})

export default function RootLayout({ children }) {
  return (
    <html lang="en" className={`${inter.variable} ${manrope.variable}`}>
      <body>{children}</body>
    </html>
  )
}
```

Benefits:
- Self-hosted fonts (no external requests)
- Automatic font subsetting
- Zero layout shift
- Preloading

Database Optimization

Indexes:

```
model Lead {
  // ...
  @@index([email])
  @@index([confirmed])
  @@index([createdAt(sort: Desc)])
}
```

Connection Pooling:

```tsx
// lib/db.ts
import { PrismaClient } from '@prisma/client'

const globalForPrisma = global as unknown as { prisma: PrismaClient }

export const db = globalForPrisma.prisma || new PrismaClient()

if (process.env.NODE_ENV !== 'production') globalForPrisma.prisma = db
```

Query Optimization:

```
// ✅ Correct (select only needed fields)
const leads = await db.lead.findMany({
  select: {
    id: true,
    name: true,
    email: true
  },
  where: {
    confirmed: true
  },
  take: 50,
  orderBy: {
    createdAt: 'desc'
  }
})

// ❌ Wrong (fetches all fields)
const leads = await db.lead.findMany()
```

🔍 SEO IMPLEMENTATION

On-Page SEO

Meta Tags (Every Page):

```
export const metadata = {
  title: 'Page Title | QuantumLeap AI',
  description: 'Compelling 150-160 character description that includes target
keywords',
  keywords: 'ai workforce, intelligent automation, cyber intelligence',
  openGraph: {
    title: 'Page Title',
    description: 'Description',
    images: ['/og-image.png'],
    type: 'website'
  },
  twitter: {
    card: 'summary_large_image',
    title: 'Page Title',
    description: 'Description',
    images: ['/og-image.png']
  },
  alternates: {
    canonical: 'https://quantumleapai.abacusai.app/page-url'
  }
}
```

Structured Data (Schema.org):

```
<script type="application/ld+json">
  {JSON.stringify({
    "@context": "https://schema.org",
    "@type": "Organization",
    "name": "QuantumLeap AI",
    "url": "https://quantumleapai.abacusai.app",
    "logo": "https://quantumleapai.abacusai.app/logo.png",
    "description": "AI-powered business transformation services",
    "contactPoint": {
      "@type": "ContactPoint",
      "telephone": "+1-555-0100",
      "contactType": "Customer Service"
    },
    "sameAs": [
      "https://twitter.com/quantumleapai",
      "https://linkedin.com/company/quantumleapai"
    ]
  })}
</script>
```

FAQ Schema:

```
<script type="application/ld+json">
  {JSON.stringify({
    "@context": "https://schema.org",
    "@type": "FAQPage",
    "mainEntity": [
      {
        "@type": "Question",
        "name": "What is AI Workforce?",
        "acceptedAnswer": {
          "@type": "Answer",
          "text": "AI Workforce replaces..."
        }
      }
    ]
  })}
</script>
```

Technical SEO

Sitemap (`public/sitemap.xml`):

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>https://quantumleapai.abacusai.app/</loc>
    <lastmod>2025-11-11</lastmod>
    <changefreq>daily</changefreq>
    <priority>1.0</priority>
  </url>
  <url>
    <loc>https://quantumleapai.abacusai.app/ai-workforce</loc>
    <lastmod>2025-11-11</lastmod>
    <changefreq>weekly</changefreq>
    <priority>0.8</priority>
  </url>
  <!-- More URLs -->
</urlset>
```

Robots.txt (`public/robots.txt`):

```
User-agent: *
Allow: /

Sitemap: https://quantumleapai.abacusai.app/sitemap.xml
```

Canonical URLs:

```
// In metadata
alternates: {
  canonical: 'https://quantumleapai.abacusai.app/page-url'
}
```

Content SEO

Keyword Strategy:
- **Primary:** ai workforce, intelligent automation, cyber intelligence, background checks
- **Secondary:** business transformation, process automation, ai employees
- **Long-tail:** “how to replace employees with ai”, “cost of bad hire calculator”

Content Guidelines:
- **Title tags:** 50-60 characters, include primary keyword
- **Meta descriptions:** 150-160 characters, include CTA
- **H1 tags:** One per page, include primary keyword
- **H2 tags:** Include secondary keywords
- **Alt text:** Descriptive, include keywords naturally
- **URL slugs:** Short, descriptive, hyphenated
- **Internal linking:** 3-5 per page to related content

Content Length:
- **Homepage:** 1,000-1,500 words
- **Service pages:** 1,500-2,000 words
- **Blog posts:** 1,500-2,500 words

Local SEO (Future)

Google Business Profile:
- Create profile for business location
- Add address, phone, hours
- Upload photos
- Collect reviews

LocalBusiness Schema:

```
{
  "@context": "https://schema.org",
  "@type": "LocalBusiness",
  "name": "QuantumLeap AI",
  "address": {
    "@type": "PostalAddress",
    "streetAddress": "123 Business St",
    "addressLocality": "City",
    "addressRegion": "State",
    "postalCode": "12345",
    "addressCountry": "US"
  },
  "geo": {
    "@type": "GeoCoordinates",
    "latitude": "40.7128",
    "longitude": "-74.0060"
  }
}
```

SEO Monitoring

Tools:
- **Google Search Console** - Track search performance
- **Google Analytics** - Track traffic sources
- **Ahrefs/SEMrush** - Track rankings
- **Screaming Frog** - Technical SEO audit

Key Metrics:
- Organic traffic
- Keyword rankings
- Click-through rate (CTR)
- Bounce rate
- Time on page
- Conversion rate

🐛 COMMON ISSUES AND SOLUTIONS

Issue 1: Build Errors

Symptom: `npm run build` fails with TypeScript errors

Causes:
- Type mismatches

- Missing imports
- Incorrect component props

Solutions:

```
# 1. Check TypeScript errors
npx tsc --noEmit

# 2. Fix type errors
# Example: Add missing type
interface ButtonProps {
  onClick: () => void
  children: React.ReactNode
}

# 3. Update imports
# Example: Use correct import path
import { Button } from '@/components/ui/button' // ✅
import { Button } from '../ui/button' // ❌
```

Issue 2: Database Connection Failures

Symptom: “Can’t reach database server” error

Causes:
- Incorrect DATABASE_URL
- Database server down
- Firewall blocking connection

Solutions:

```
# 1. Verify DATABASE_URL
echo $DATABASE_URL

# 2. Test connection
npx prisma db pull

# 3. Check Supabase dashboard
# Ensure database is running

# 4. Regenerate Prisma client
npx prisma generate
```

Issue 3: API Routes Returning 500 Errors

Symptom: API requests fail with 500 Internal Server Error

Causes:
- Unhandled exceptions
- Missing environment variables
- Database query errors

Solutions:

```
// Add try-catch to all API routes
export async function POST(request: NextRequest) {
  try {
    const data = await request.json()
    // Process data
    return NextResponse.json({ success: true })
  } catch (error) {
    console.error('API Error:', error)
    return NextResponse.json(
      { error: 'Internal server error' },
      { status: 500 }
    )
  }
}

// Log errors for debugging
import { sendDiscordNotification } from '@/lib/discord'

catch (error) {
  console.error('API Error:', error)

  // Alert team via Discord
  await sendDiscordNotification({
    title: '🚨 API Error',
    description: error.message,
    color: 0xff0000
  })

  return NextResponse.json({ error: 'Error' }, { status: 500 })
}
```

Issue 4: Email Not Sending

Symptom: Confirmation emails not received

Causes:
- Gmail API authentication failure
- Invalid email address
- Email in spam folder

Solutions:

```
# 1. Verify Gmail API credentials
# Check .env.local has correct:
GMAIL_CLIENT_EMAIL
GMAIL_PRIVATE_KEY
GMAIL_SENDER

# 2. Test email sending
# Create test script
node test-email.js

# 3. Check Gmail API quota
# Go to Google Cloud Console
# APIs & Services > Dashboard
# Check quota usage

# 4. Verify service account permissions
# Ensure domain-wide delegation enabled
```

Issue 5: Prisma Client Out of Sync

Symptom: “Prisma client doesn’t match schema” error

Cause: Schema changed but client not regenerated

Solution:

```
# Regenerate Prisma client
npx prisma generate

# If still failing, force regenerate
rm -rf node_modules/.prisma
npx prisma generate
```

Issue 6: Hydration Errors

Symptom: “Hydration failed” error in console

Causes:
- Server HTML doesn’t match client HTML
- Using `Date.now()` or `Math.random()` during render
- Browser extensions modifying DOM

Solutions:

```
// ❌ Wrong (causes hydration error)
function Component() {
  const timestamp = Date.now()
  return <div>{timestamp}</div>
}

// ✅ Correct (client-side only)
'use client'
import { useState, useEffect } from 'react'

function Component() {
  const [timestamp, setTimestamp] = useState<number | null>(null)

  useEffect(() => {
    setTimestamp(Date.now())
  }, [])

  return <div>{timestamp || 'Loading...'}</div>
}
```

Issue 7: Slow Page Load Times

Symptom: Pages take > 3 seconds to load

Causes:
- Large images not optimized
- Too many external requests
- Blocking JavaScript

Solutions:

```
# 1. Analyze with Lighthouse
lighthouse https://quantumleapai.abacusai.app

# 2. Optimize images
# Use next/image component
# Compress images (TinyPNG)

# 3. Lazy load components
import dynamic from 'next/dynamic'
const Heavy = dynamic(() => import('./Heavy'))

# 4. Enable caching
# Add Cache-Control headers
```

Issue 8: Calculator Not Submitting

Symptom: Calculator form submits but no response

Causes:
- API route error
- Network issue
- Frontend error handling missing

Solutions:

```
// Add error handling
const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault()
  setLoading(true)
  setError(null)

  try {
    const response = await fetch('/api/calculator/recommendations', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(formData)
    })

    if (!response.ok) {
      throw new Error('API request failed')
    }

    const data = await response.json()
    setResults(data)
  } catch (error) {
    console.error('Submit error:', error)
    setError('Failed to calculate. Please try again.')
  } finally {
    setLoading(false)
  }
}
```

Issue 9: Admin Login Not Working

Symptom: “Invalid credentials” even with correct password

Causes:
- Password hash mismatch

- JWT secret missing
- Cookie not being set

Solutions:

```
# 1. Verify admin user exists
npx prisma studio
# Check admin_users table

# 2. Reset admin password
npx tsx scripts/reset-admin-password.ts

# 3. Verify JWT_SECRET set
echo $ADMIN_JWT_SECRET

# 4. Check browser cookies
# DevTools > Application > Cookies
# Verify admin_token present
```

Issue 10: Discord Notifications Not Sending

Symptom: Leads captured but no Discord notification

Causes:
- Invalid webhook URL
- Network error
- Webhook rate limited

Solutions:

```
# 1. Verify webhook URL
echo $DISCORD_WEBHOOK_URL

# 2. Test webhook
curl -X POST $DISCORD_WEBHOOK_URL \
  -H "Content-Type: application/json" \
  -d '{"content": "Test message"}'

# 3. Check Discord server
# Verify webhook still exists
# Webhooks can be deleted by accident

# 4. Add retry logic
async function sendDiscordNotification(data, retries = 3) {
  for (let i = 0; i < retries; i++) {
    try {
      await fetch(DISCORD_WEBHOOK_URL, {
        method: 'POST',
        body: JSON.stringify(data)
      })
      return
    } catch (error) {
      if (i === retries - 1) throw error
      await new Promise(r => setTimeout(r, 1000 * (i + 1)))
    }
  }
}
```

🛠️ DEVELOPMENT WORKFLOW

Git Workflow

Branches:
- `master` - Main development branch
- `semi-final` - Production branch (current)
- `feature/*` - Feature branches
- `hotfix/*` - Urgent fixes

Commit Convention:

```
<type>(<scope>): <subject>

<body>

<footer>
```

Types:
- `feat` - New feature
- `fix` - Bug fix
- `docs` - Documentation
- `style` - Code style (formatting)
- `refactor` - Code refactoring
- `test` - Adding tests
- `chore` - Maintenance

Examples:

```
git commit -m "feat(calculator): Add AI Team Calculator"
git commit -m "fix(email): Fix Gmail API authentication"
git commit -m "docs(readme): Update installation instructions"
```

Code Review Checklist

Before Submitting PR:
- [ ] Code follows project style guide
- [ ] All TypeScript errors resolved
- [ ] ESLint warnings addressed
- [ ] Build completes successfully
- [ ] Tested locally (desktop + mobile)
- [ ] No console errors
- [ ] Documentation updated (if needed)

Reviewer Checklist:
- [ ] Code quality (readable, maintainable)
- [ ] Security concerns addressed
- [ ] Performance implications considered
- [ ] Test coverage adequate (when tests exist)
- [ ] Breaking changes documented

Code Style Guide

TypeScript:

```
// ✅ Use explicit types
function calculateROI(salary: number, employees: number): number {
  return (salary * employees) - 5000
}

// ❌ Avoid 'any' type
function calculate(data: any) {
  return data.value
}

// ✅ Use interfaces for objects
interface User {
  id: string
  name: string
  email: string
}

// ✅ Use const for non-changing values
const API_URL = 'https://api.example.com'

// ✅ Use meaningful variable names
const monthlyRevenue = 10000 // ✅
const mr = 10000 // ❌
```

React/Next.js:

```
// ✅ Use functional components
export function Button({ children }: { children: React.ReactNode }) {
  return <button>{children}</button>
}

// ✅ Use 'use client' directive when needed
'use client'
import { useState } from 'react'

// ✅ Extract reusable logic to hooks
function useWindowSize() {
  const [size, setSize] = useState({ width: 0, height: 0 })
  // ... logic
  return size
}

// ✅ Keep components small and focused
// If > 200 lines, consider breaking into smaller components
```

Naming Conventions:

```
Files:
- kebab-case for files: user-profile.tsx
- PascalCase for components: UserProfile.tsx

Variables:
- camelCase for variables: const userName = 'John'
- PascalCase for types/interfaces: interface UserProfile {}
- SCREAMING_SNAKE_CASE for constants: const API_KEY = 'xxx'

Functions:
- camelCase for functions: function calculateTotal() {}
- Prefix with 'handle' for event handlers: handleClick()
- Prefix with 'is/has' for booleans: isLoading, hasPermission
```

Development Tools

VS Code Extensions:
- ESLint
- Prettier
- Tailwind CSS IntelliSense
- Prisma
- GitLens
- Error Lens
- Auto Rename Tag
- Import Cost

Browser Extensions:
- React Developer Tools
- Redux DevTools (if using Redux)
- Lighthouse
- JSON Viewer

Command Line Tools:
- `npm-check-updates` - Update dependencies
- `serve` - Test production build locally
- `pm2` - Process manager
- `ngrok` - Expose local server

Debugging Tips

React DevTools:

```
# View component tree
# Inspect props and state
# Trace component updates
# Profile performance
```

Console Logging:

```
// ✅ Structured logging
console.log('User data:', { user, timestamp: Date.now() })

// ✅ Use console methods
console.warn('Deprecated function')
console.error('API Error:', error)
console.table(arrayOfObjects)

// ❌ Remove console.logs before committing
// Use environment variable instead
if (process.env.NODE_ENV === 'development') {
  console.log('Debug info')
}
```

Network Debugging:

```
# Chrome DevTools > Network tab
# Filter by type (XHR, Fetch)
# Check request/response headers
# Inspect payload
# Check timing (TTFB, download)
```

Database Debugging:

```
# Use Prisma Studio
npx prisma studio

# Log SQL queries
# Enable in schema.prisma
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

# Then check logs for generated SQL
```

🔒 SECURITY BEST PRACTICES

Environment Variables

✅ **Do:**
- Store secrets in `.env.local` (gitignored)
- Use different values for dev/prod
- Rotate secrets regularly
- Use strong, random values

❌ **Don't:**
- Commit `.env` files
- Expose secrets in client-side code
- Hard-code secrets in code
- Share secrets via email/chat

API Security

Authentication:

```
// Verify JWT token on protected routes
import { verifyToken } from '@/lib/admin-auth'

export async function GET(request: NextRequest) {
  const token = request.cookies.get('admin_token')?.value

  if (!token) {
    return NextResponse.json({ error: 'Unauthorized' }, { status: 401 })
  }

  try {
    const payload = verifyToken(token)
    // Process request
  } catch (error) {
    return NextResponse.json({ error: 'Invalid token' }, { status: 401 })
  }
}
```

Rate Limiting:

```
// Implement rate limiting
import { RateLimiter } from '@/lib/rate-limiter'

const limiter = new RateLimiter({
  interval: 60 * 1000, // 1 minute
  uniqueTokenPerInterval: 500
})

export async function POST(request: NextRequest) {
  const ip = request.ip || 'anonymous'

  try {
    await limiter.check(ip, 10) // 10 requests per minute
  } catch {
    return NextResponse.json(
      { error: 'Rate limit exceeded' },
      { status: 429 }
    )
  }

  // Process request
}
```

Input Validation:

```
// Validate all user inputs
import { z } from 'zod'

const leadSchema = z.object({
  name: z.string().min(2).max(100),
  email: z.string().email(),
  company: z.string().max(100).optional(),
  phone: z.string().regex(/^\+?[1-9]\d{1,14}$/).optional()
})

export async function POST(request: NextRequest) {
  const body = await request.json()

  try {
    const validatedData = leadSchema.parse(body)
    // Process valid data
  } catch (error) {
    return NextResponse.json(
      { error: 'Invalid input' },
      { status: 400 }
    )
  }
}
```

SQL Injection Prevention:

```
// ✅ Correct (Prisma prevents SQL injection)
const user = await db.user.findUnique({
  where: { email: userInput }
})

// ❌ Wrong (vulnerable to SQL injection)
const user = await db.$queryRaw`SELECT * FROM users WHERE email = ${userInput}`
```

XSS Prevention:

```
// ✅ React auto-escapes by default
<div>{userInput}</div>

// ❌ Dangerous (only use for trusted content)
<div dangerouslySetInnerHTML={{ __html: userInput }} />

// ✅ Sanitize if you must use dangerouslySetInnerHTML
import DOMPurify from 'dompurify'
<div dangerouslySetInnerHTML={{ __html: DOMPurify.sanitize(userInput) }} />
```

Password Security

Hashing:

```
import bcrypt from 'bcryptjs'

// ✅ Hash password before storing
const passwordHash = await bcrypt.hash(password, 10) // 10 rounds

// ✅ Verify password
const valid = await bcrypt.compare(password, storedHash)

// ❌ Never store plaintext passwords
const user = await db.user.create({
  data: { password: plaintextPassword } // ❌ WRONG
})
```

Password Requirements:

- Minimum 8 characters
- At least one uppercase letter
- At least one lowercase letter
- At least one number
- At least one special character

```
const passwordSchema = z.string()
  .min(8, 'Password must be at least 8 characters')
  .regex(/[A-Z]/, 'Password must contain at least one uppercase letter')
  .regex(/[a-z]/, 'Password must contain at least one lowercase letter')
  .regex(/[0-9]/, 'Password must contain at least one number')
  .regex(/[^A-Za-z0-9]/, 'Password must contain at least one special character')
```

CORS Configuration

```
// next.config.js
module.exports = {
  async headers() {
    return [
      {
        source: '/api/:path*',
        headers: [
          { key: 'Access-Control-Allow-Origin', value: 'https://quantum-
leapai.abacusai.app' },
          { key: 'Access-Control-Allow-Methods', value:
'GET,POST,PUT,DELETE,OPTIONS' },
          { key: 'Access-Control-Allow-Headers', value: 'Content-Type, Authoriza-
tion' }
        ]
      }
    ]
  }
}
```

Content Security Policy

```
// next.config.js
module.exports = {
  async headers() {
    return [
      {
        source: '/:path*',
        headers: [
          {
            key: 'Content-Security-Policy',
            value: `
              default-src 'self';
              script-src 'self' 'unsafe-eval' 'unsafe-inline';
              style-src 'self' 'unsafe-inline';
              img-src 'self' data: https:;
              font-src 'self' data:;
              connect-src 'self' https://ilvxbllftqzewiojsryq.supabase.co;
            `.replace(/\s{2,}/g, ' ').trim()
          }
        ]
      }
    ]
  }
}
```

📊 MONITORING AND ANALYTICS

Google Analytics (Future)

Setup:

```
npm install @next/third-parties
```

Implementation:

```
// app/layout.tsx
import { GoogleAnalytics } from '@next/third-parties/google'

export default function RootLayout({ children }) {
  return (
    <html>
      <body>
        {children}
        <GoogleAnalytics gaId="G-XXXXXXXXXX" />
      </body>
    </html>
  )
}
```

Track Events:

```
// Track calculator submissions
window.gtag('event', 'calculator_submission', {
  calculator_type: 'ai-team',
  estimated_savings: 225000
})

// Track consultation bookings
window.gtag('event', 'consultation_booking', {
  service: 'AI Workforce',
  value: 5000
})
```

Error Monitoring (Future)

Sentry Setup:

```
npm install @sentry/nextjs
```

Configuration:

```
// sentry.client.config.ts
import * as Sentry from '@sentry/nextjs'

Sentry.init({
  dsn: process.env.NEXT_PUBLIC_SENTRY_DSN,
  tracesSampleRate: 1.0,
  environment: process.env.NODE_ENV
})
```

Performance Monitoring

Web Vitals:

```
// app/layout.tsx
import { Analytics } from '@vercel/analytics/react'

export default function RootLayout({ children }) {
  return (
    <html>
      <body>
        {children}
        <Analytics />
      </body>
    </html>
  )
}
```

Custom Logging

Logger Utility:

```
// lib/logger.ts
export const logger = {
  info: (message: string, data?: any) => {
    console.log(`[INFO] ${message}`, data)
  },
  warn: (message: string, data?: any) => {
    console.warn(`[WARN] ${message}`, data)
  },
  error: (message: string, error?: any) => {
    console.error(`[ERROR] ${message}`, error)

    // Send to error tracking service
    // await sendToSentry(error)
  }
}
```

🗺️ FUTURE ROADMAP

Phase 1: Enhancements (1-3 Months)

Features:
- [ ] User authentication (customer portal)
- [ ] Project dashboard for clients
- [ ] Real-time chat support
- [ ] Advanced analytics dashboard
- [ ] A/B testing framework
- [ ] Email marketing integration (Mailchimp/SendGrid)

Technical:
- [ ] Unit test coverage (80%+)
- [ ] E2E test suite (Playwright)
- [ ] CI/CD pipeline (GitHub Actions)
- [ ] Automated deployment
- [ ] Database backup automation

Phase 2: Growth (3-6 Months)

Features:
- [ ] Multi-language support (i18n)
- [ ] Advanced CRM features
- [ ] Payment processing (Stripe)
- [ ] Subscription management
- [ ] Customer self-service portal
- [ ] Knowledge base/Help center

Marketing:
- [ ] SEO content strategy (100+ blog posts)
- [ ] Social media integration
- [ ] Referral program
- [ ] Affiliate program
- [ ] Webinar/event booking

Phase 3: Scale (6-12 Months)

Features:
- [ ] Mobile app (React Native)
- [ ] Advanced AI chatbot
- [ ] Video consultation platform
- [ ] Custom reporting engine
- [ ] API for third-party integrations
- [ ] White-label solution

Infrastructure:
- [ ] Microservices architecture
- [ ] Kubernetes deployment
- [ ] Multi-region hosting
- [ ] CDN optimization
- [ ] Advanced caching strategy

📞 SUPPORT AND CONTACTS

Key Personnel

Founder & CEO:
- **Name:** Paras Khurana
- **Email:** paras@leapintoai.com
- **Role:** Business strategy, client relations

Technical Lead:
- **Name:** [To be assigned]
- **Email:** [To be assigned]
- **Role:** Technical decisions, architecture

Support Email:
- **Email:** support@leapintoai.com
- **Purpose:** General support inquiries

External Resources

Hosting & Infrastructure:
- **Platform:** Abacus.AI
- **Dashboard:** https://apps.abacus.ai
- **Support:** support@abacus.ai

Database:
- **Provider:** Supabase
- **Dashboard:** https://supabase.com/dashboard
- **Documentation:** https://supabase.com/docs

Email API:
- **Provider:** Gmail API (Google Cloud)
- **Console:** https://console.cloud.google.com
- **Documentation:** https://developers.google.com/gmail/api

Booking:
- **Provider:** TidyCal
- **Dashboard:** https://tidycal.com/dashboard
- **API Docs:** https://tidycal.com/api

Documentation Links

Framework & Libraries:
- Next.js: https://nextjs.org/docs
- React: https://react.dev
- TypeScript: https://www.typescriptlang.org/docs
- Tailwind CSS: https://tailwindcss.com/docs
- Prisma: https://www.prisma.io/docs
- Framer Motion: https://www.framer.com/motion

UI Components:
- shadcn/ui: https://ui.shadcn.com
- Radix UI: https://www.radix-ui.com

Emergency Contacts

Critical Issues (Production Down):
1. Check Abacus.AI dashboard status
2. Check Supabase dashboard status
3. Contact support@abacus.ai
4. Contact paras@leapintoai.com

Database Issues:
1. Check Supabase dashboard
2. Review recent migrations
3. Check connection pool limits
4. Restore from backup if needed

Email Delivery Issues:
1. Check Gmail API quota
2. Verify service account permissions
3. Test with SMTP backup (Hostinger)
4. Contact Google Cloud support

✅ HANDOVER CHECKLIST

Knowledge Transfer

- [ ] Read both parts of this handover document
- [ ] Review project structure
- [ ] Understand design system (Quantum Gradient Dark)
- [ ] Familiarize with database schema
- [ ] Review API endpoints
- [ ] Understand lead capture flow
- [ ] Review calculator logic
- [ ] Understand admin dashboard

Access & Permissions

- [ ] GitHub repository access granted
- [ ] Abacus.AI dashboard access
- [ ] Supabase dashboard access
- [ ] Google Cloud Console access
- [ ] TidyCal account access
- [ ] Discord server access (for notifications)
- [ ] Admin login credentials provided

Environment Setup

- [ ] Clone repository
- [ ] Install dependencies
- [ ] Set up environment variables
- [ ] Configure database connection
- [ ] Test local development server
- [ ] Verify build process works
- [ ] Test admin dashboard login

Testing

- [ ] Test homepage functionality
- [ ] Test all calculators
- [ ] Test lead capture flow
- [ ] Test email sending
- [ ] Test consultation booking
- [ ] Test admin dashboard features
- [ ] Test on multiple browsers
- [ ] Test on mobile devices

Deployment

- [ ] Understand deployment process
- [ ] Verify production environment variables
- [ ] Test staging deployment
- [ ] Monitor production deployment
- [ ] Verify all features work in production

Documentation

- [ ] Update documentation as needed
- [ ] Document any changes made
- [ ] Create runbook for common issues
- [ ] Document any custom configurations

🎉 CONCLUSION

This document provides a complete technical overview of the QuantumLeap AI website project. It covers everything from architecture and implementation to deployment and maintenance.

Key Takeaways:
1. **Modern Stack** - Next.js 14, TypeScript, Tailwind CSS, Prisma
2. **Production-Ready** - Fully functional and deployed
3. **Well-Documented** - 15+ technical documents
4. **Scalable** - Ready for growth and enhancements
5. **Maintainable** - Clean code, clear structure

Next Steps:
1. Set up your local environment
2. Test all features locally
3. Make a small change and deploy
4. Join Discord for team communication
5. Review the future roadmap

Questions?
- Email: paras@leapintoai.com
- Support: support@leapintoai.com

Welcome to the team! 🚀

Document End

Last Updated: November 11, 2025
Version: 2.0
Author: DeepAgent AI
Project: QuantumLeap AI Website
Repository: https://github.com/AICyberSecurity911/WBsite.git