8-Week Development Plan - Corrected & Implementation-Ready

Date: January 26, 2025
Status: Corrected Based on Codebase Analysis
Confidence: 95% (pending timeline confirmation)

Key Corrections Made

1. ✅ **Schema Alignment:** Updated to match existing `docs/SUPABASE_CASE_STUDIES_SCHEMA.md`
2. ✅ **ID Types:** All tables use UUID (consistent with existing tables)
3. ✅ **Auth Status:** Documented existing middleware (needs role-based enhancement)
4. ✅ **PDF Method:** Confirmed VS Code extension (matches existing workflow)
5. ✅ **API Comments:** Noted need to update "Claude 3.5 Sonnet" → "Claude 3 Haiku"

Week 2: Integration Testing & Refinement

Dates: [ADJUST BASED ON TIMELINE CONFIRMATION] | Status: READY TO START

2.1: Manual Testing & Validation (3 hours)

Goal: Verify all systems work as expected across devices

Files to Test (DO NOT MODIFY):

- `app/api/ai-assistant/route.ts`
- `components/AIAssistantWidget.tsx`
- `app/layout.tsx`

Testing Checklist:

- Desktop: Chrome, Firefox, Safari
- Mobile: iPhone, Android
- Error scenarios: Network disconnect, rate limits, invalid input
- Integration: Widget on all pages, Supabase storage, no console errors

Output: Testing report documenting all verified functionality

2.2: Performance Optimization (2 hours)

Goal: Ensure chat widget doesn't impact page load performance

Files to Analyze (READ ONLY):

- `app/layout.tsx` (widget loading)
- `components/AIAssistantWidget.tsx` (component optimization)

Performance Checks:

- Lighthouse score (target: maintain 91-98/100)
- Load time metrics (FCP, LCP, CLS, TTI)
- Bundle size analysis

Modifications Allowed:

- Add React.memo() if needed
- Lazy load widget if necessary
- Optimize imports (only if impacts performance)

DO NOT CHANGE:

- Model (claude-3-haiku-20240307) ✅
- API endpoint ✅
- Supabase integration ✅
- Error handling ✅

Output: Performance report with Lighthouse scores

2.3: Documentation Review & Updates (2 hours)

Goal: Ensure documentation matches current implementation

Files to Review:

- `docs/AI_ASSISTANT_WIDGET_DEPLOYMENT.md` (if exists)
- `docs/AI_ASSISTANT_TESTING_GUIDE.md` (if exists)
- `docs/LIGHTHOUSE_AUDIT_GUIDE.md`
- `docs/CASE_STUDY_PDF_GUIDE.md`
- `README.md`

Documentation Updates:

- Update deployment guide with Haiku model info
- Document Claude 3 Haiku (current model)
- Note: Sonnet requires plan upgrade
- Update troubleshooting for model access (402, 403, 429 errors)
- Document credit verification steps

Create New Documentation:

- `docs/DEPLOYMENT_CHECKLIST.md`
- `docs/MODEL_SELECTION_GUIDE.md` (Haiku vs Sonnet)
- `docs/ARCHITECTURE_OVERVIEW.md` (system diagram)

Output: Updated documentation ready for sharing

Note: Fix API route comment in `app/api/ai-assistant/route.ts` line 6: "Claude 3.5 Sonnet" → "Claude 3 Haiku"

Week 3: Second Case Study - AI Assistant

Dates: [ADJUST BASED ON TIMELINE] | Status: READY TO PLAN

3.1: Planning & Documentation (2 hours)

Goal: Create comprehensive case study for AI Assistant feature

Research Files (REFERENCE ONLY):

- `app/api/ai-assistant/route.ts` (implementation details)
- `components/AIAssistantWidget.tsx` (widget features)
- `lib/claude.ts` (Claude integration)
- `supabase/migrations/20250125_create_ai_assistant_tables.sql` (schema design)

Case Study Outline:

1. Challenge (customer support bottleneck, 24/7 availability)
2. Approach (Claude 3 Haiku selection, architecture design)
3. Implementation (Next.js, TypeScript, Claude API, Supabase, Vercel)
4. Results (real-time streaming, persistent storage, production metrics)
5. Technical Details (model selection, response time, cost analysis)

6. Code Examples (widget, API endpoint, error handling)
7. ROI Analysis (development time, monthly cost, lead capture)

Output: Case study outline ready for development

3.2: Case Study Development (4 hours)

Goal: Write complete case study (1,200+ words)

Files to Reference (READ ONLY):

- Week 1 completion documentation
- `docs/WEEK_1_COMPLETION_SUMMARY.md`
- `docs/AI_ASSISTANT_TESTING_GUIDE.md` (if exists)

Case Study Contents:

- Header & Intro (150 words)
- Challenge Section (200 words)
- Approach Section (250 words)
- Implementation Section (300 words)
- Results Section (200 words)
- Learnings & Insights (150 words)
- Technical Appendix (150 words)

New Files to Create:

- `case-studies/ai-assistant/case-study.md` (1,200+ words)
- `case-studies/ai-assistant/README.md`
- `case-studies/ai-assistant/screenshot-chat.png` (widget screenshot)
- `case-studies/ai-assistant/diagram-architecture.png` (system diagram)

Output: Complete case study + screenshots + diagrams

3.3: Case Study PDF Generation (1 hour)

Goal: Generate professional PDF for submission

Method: VS Code Extension (confirmed working method)

1. Install "Markdown PDF" extension (by yzane)
2. Right-click `case-studies/ai-assistant/case-study.md`
3. Select "Markdown PDF: Export (pdf)"
4. Save as `case-studies/ai-assistant/case-study.pdf`

Verification:

- PDF generated successfully
- All content included
- Formatting looks professional
- Images embedded properly

Output: `case-studies/ai-assistant/case-study.pdf`

Week 4: Case Studies Database & Admin UI

Dates: [ADJUST BASED ON TIMELINE] | Status: PLANNING PHASE

4.1: Database Schema Design (2 hours)

Goal: Design case studies database schema

⚠️ **CRITICAL:** Use existing schema from `docs/SUPABASE_CASE_STUDIES_SCHEMA.md` (NOT the plan's original schema)

DO NOT TOUCH: Existing Supabase setup (ai_assistant_* tables)

Migration File: `supabase/migrations/YYYYMMDD_create_case_studies_table.sql`

Schema to Use:

- UUID PRIMARY KEY (not SERIAL)
- JSONB for content and metrics (flexible structure)
- Full-text search index
- Comprehensive RLS policies
- Includes: featured, published_at, category, etc.

Reference: `docs/SUPABASE_CASE_STUDIES_SCHEMA.md` (lines 23-80)

Security:

- Enable RLS (Row Level Security)
- Public read access (published = true)
- Admin-only write access
- Service role access for API operations

Output: Supabase migration SQL file matching existing schema documentation

4.2: Admin Dashboard - Case Studies Management (5 hours)

Goal: Build admin UI for managing case studies

⚠️ **AUTHENTICATION NOTE:**

- Middleware exists but only checks authentication (not admin role)
- **Recommendation:** Add role-based access control before Week 4
- Update `middleware.ts` to check `session.user.role === 'admin'`

New Files to Create:

- `app/admin/case-studies/page.tsx` (list view)
- `app/admin/case-studies/[id]/page.tsx` (edit view)
- `app/admin/case-studies/new/page.tsx` (create view)
- `components/AdminCaseStudyForm.tsx`
- `lib/supabase/case-studies.ts` (database operations - use existing pattern)

Admin Features:

- List view: Display all, filter, sort, quick edit
- Create/Edit view: Form with rich text, image upload, metrics input
- Authentication: Admin-only access (add role check to middleware)
- Preview: Draft vs published toggle

DO NOT BREAK:

- AI Assistant widget
- Accessibility compliance (WCAG 2.1 AA)
- Lighthouse score (>91)
- Existing database tables

Output: Admin UI fully functional with CRUD operations

4.3: Public Case Studies Display (3 hours)

Goal: Display case studies on public website

New Files to Create:

- `app/case-studies/page.tsx` (list of all case studies)
- `app/case-studies/[slug]/page.tsx` (individual case study)
- `components/CaseStudyCard.tsx` (card component)
- `components/CaseStudyGrid.tsx` (grid layout)

Features:

- Case Studies Page: Grid, filter, search, responsive, SEO optimized
- Individual Case Study Page: Full content, metrics, images, PDF download, related studies, CTA

Use Existing Pattern:

- Reference: `lib/supabase/case-studies.ts` (from docs/SUPABASE_CASE_STUDIES_SCHEMA.md)
- Server components for data fetching
- Client components for interactivity

Preservation:

- WCAG 2.1 AA compliance
- Lighthouse score maintenance
- Widget functionality
- AI Assistant on all pages

Output: Case studies publicly accessible and beautiful

Week 5: Testimonials & Social Proof

Dates: [ADJUST BASED ON TIMELINE] | Status: PLANNING PHASE

5.1: Testimonials Database & Admin (2 hours)

Goal: Add customer testimonials system

Migration File: `supabase/migrations/YYYYMMDD_create_testimonials_table.sql`

Schema (CORRECTED - Uses UUID):

```sql
CREATE TABLE testimonials (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  client_name TEXT NOT NULL,
  client_title TEXT,
  client_company TEXT,
  client_image_url TEXT,
  content TEXT NOT NULL,
  rating INT CHECK (rating >= 1 AND rating <= 5),
  service_type TEXT,
  case_study_id UUID REFERENCES case_studies(id), -- Optional link to case study
  published BOOLEAN DEFAULT false,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Indexes
CREATE INDEX idx_testimonials_published ON testimonials(published) WHERE published = true;
CREATE INDEX idx_testimonials_created_at ON testimonials(created_at DESC);
CREATE INDEX idx_testimonials_case_study_id ON testimonials(case_study_id);
```

```sql
-- RLS Policies
ALTER TABLE testimonials ENABLE ROW LEVEL SECURITY;

CREATE POLICY "Public can read published testimonials"
  ON testimonials FOR SELECT
  USING (published = true);

CREATE POLICY "Admins can manage testimonials"
  ON testimonials FOR ALL
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM auth.users
      WHERE auth.users.id = auth.uid()
      AND auth.users.raw_user_meta_data->>'role' = 'admin'
    )
  );
```

New Files:

- `app/admin/testimonials/page.tsx`
- `lib/admin-testimonials.ts` (use `createServerSupabaseClient(true)` pattern)

Output: Testimonials database with admin UI

5.2: Testimonials Display (2 hours)

Goal: Display testimonials on website

New Files:

- `components/TestimonialCarousel.tsx`
- `components/TestimonialCard.tsx`
- `app/testimonials/page.tsx` (optional testimonials page)

Features:

- Carousel/slider display
- Star rating display
- Client image & name
- Responsive design
- WCAG 2.1 AA accessible
- Auto-rotate or manual controls

Output: Testimonials displayed on website

5.3: Case Study - Testimonials Feature (2 hours)

Goal: Create third case study (testimonials integration)

Output: `case-studies/testimonials/case-study.pdf`

Week 6: Contact Forms & Lead Capture

Dates: [ADJUST BASED ON TIMELINE] | Status: PLANNING PHASE

6.1: Enhanced Contact Form (2 hours)

Goal: Improve contact form with validation & storage

Update/Create Files:

- `app/api/contact/route.ts` (enhance if exists, or create new)
- Contact form component

Features:

- Form validation (client & server)
- Honeypot field (spam prevention)
- Rate limiting (prevent abuse)
- Supabase storage (leads table)
- Email notification (to you)
- Auto-reply (to customer)
- Success message

Migration File: `supabase/migrations/YYYYMMDD_create_leads_table.sql`

Schema (CORRECTED - Uses UUID, TIMESTAMPTZ):

```
CREATE TABLE leads (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name TEXT NOT NULL,
  email TEXT NOT NULL,
  phone TEXT,
  company TEXT,
  project_type TEXT,
  message TEXT,
  source TEXT, -- 'contact form', 'chat widget', etc.
  status TEXT DEFAULT 'new' CHECK (status IN ('new', 'contacted', 'qualified', 'converted', 'lost')),
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Indexes
CREATE INDEX idx_leads_email ON leads(email);
CREATE INDEX idx_leads_status ON leads(status);
CREATE INDEX idx_leads_created_at ON leads(created_at DESC);
CREATE INDEX idx_leads_source ON leads(source);

-- RLS Policies (admin-only access)
ALTER TABLE leads ENABLE ROW LEVEL SECURITY;

CREATE POLICY "Admins can manage leads"
  ON leads FOR ALL
  TO authenticated
  USING (
    EXISTS (
      SELECT 1 FROM auth.users
      WHERE auth.users.id = auth.uid()
      AND auth.users.raw_user_meta_data->>'role' = 'admin'
    )
  );
```

Output: Enhanced contact form with lead storage

6.2: Lead Management Admin (2 hours)

Goal: Admin UI for managing leads

New Files:

- `app/admin/leads/page.tsx`
- `lib/admin-leads.ts` (use `createServerSupabaseClient(true)` pattern)

Features:

- Lead list with filters
- Status management (new, contacted, qualified, converted, lost)
- Search by name/email
- Bulk export to CSV (optional)
- Contact history tracking

Output: Lead management admin UI

6.3: Integration with AI Chat Widget (1 hour)

Goal: Capture leads from chat widget

⚠️ **CRITICAL:** Update file carefully - do not break chat functionality

Update File:

- `components/AIAssistantWidget.tsx`

Additions:

- Capture user email if provided in conversation
- Store in leads table (async, non-blocking)
- Mark source as "chat widget"
- Don't break chat functionality

Verification After Change:

- `npm run dev`
- Test `/test-ai` page
- Send message (verify works)
- Check leads in Supabase
- No TypeScript errors
- No console errors

Output: Chat widget captures leads

Week 7: Government Tender Submission

Dates: [ADJUST BASED ON TIMELINE] | Status: PLANNING PHASE

7.1: Tender Preparation Package (2 hours)

Goal: Compile everything needed for government tender

Package Contents:

- Accessibility Proof (Lighthouse audit, WCAG 2.1 AA compliance)
- Case Studies (3 completed PDFs: accessibility, AI assistant, testimonials)
- Technical Documentation (architecture, stack, security, deployment)
- Company Information (profile, experience, services, pricing)
- References & Proof (GitHub link, live URL, metrics, certifications)

Output: Tender submission package ready

7.2: Tender Submission (1 hour)

Goal: Submit to government opportunities

Opportunities to Research:

- Queensland government contracts
- Australian government contracts
- Local council digital initiatives
- Agency tender listings
- Panel registration (whole-of-government)

Output: Tender applications submitted

Week 8: Optimization & Scaling

Dates: [ADJUST BASED ON TIMELINE] | Status: PLANNING PHASE

8.1: Performance Optimization (3 hours)

Goal: Ensure optimal performance at scale

Areas to Optimize:

- Lighthouse Score (target: 95+, currently 91-98)
- Chat Widget Performance (response time < 1s, streaming latency)
- SEO (meta tags, structured data, sitemap, robots.txt)

Files to Review (READ ONLY):

- `app/layout.tsx`
- All `app/**/page.tsx` files
- Components (for images)

Modifications Allowed:

- Add meta tags
- Optimize images (Next.js Image component)
- Add lazy loading
- Code splitting

DO NOT BREAK:

- AI Assistant widget
- Accessibility compliance
- Existing functionality

Output: Optimized performance (95+ Lighthouse)

8.2: Analytics & Monitoring (2 hours)

Goal: Track usage and performance

Implementation:

- Analytics: Page views, chat widget usage, lead capture source, case study views
- Monitoring: Error tracking (Sentry), performance monitoring, uptime monitoring
- New Integrations: Google Analytics 4, error tracking (optional)

Output: Analytics dashboard tracking all metrics

8.3: Scale Planning (1 hour)

Goal: Plan for growth

Scalability Considerations:

- Database: Supabase auto-scaling, indexes, connection pooling, backups
- API: Rate limiting, validation, error handling, logging
- Frontend: Code splitting, image optimization, caching, CDN
- Infrastructure: Vercel auto-scaling, database backups, disaster recovery, security audit

Output: Scaling strategy documented

Critical Protection Rules

Files You MUST NOT BREAK

1. **app/api/ai-assistant/route.ts**
   - Changes = broken chat
   - Test: `npm run dev` → `/test-ai`
   - Verify before pushing
2. **components/AIAssistantWidget.tsx**
   - Changes = widget doesn't work
   - Test: `npm run dev` → all pages
   - Verify widget visible and functional
3. **lib/claude.ts**
   - Changes = no AI responses
   - Model MUST stay: `claude-3-haiku-20240307`
   - Error handling MUST stay intact
4. **app/layout.tsx**
   - Widget imported here
   - WCAG compliance critical
   - Lighthouse score impact
   - Test after any changes
5. **Supabase Configuration**
   - Database migrations (NEVER revert)
   - RLS policies (security)
   - Types (TypeScript safety)
   - Environment variables

Before Any Commit

Mandatory Checklist:

- `npm run dev` (local testing)
- Browser testing (widget appears)
- Chat test (send message, get response)
- TypeScript check (`npm run build`)
- No console errors (F12 → Console)
- Lighthouse score (still >91)
- WCAG compliance (no new violations)
- Git status (what's changing?)
- Commit message (clear description)

Success Metrics

Week 1 (COMPLETED ✅)

- 6 accessibility violations fixed (0 remaining)
- Lighthouse: 91-98/100
- WCAG 2.1 AA compliant
- AI Assistant implemented
- Chat widget live
- Build passing (0 errors)
- TypeScript: 100% type-safe

Week 2 (NEXT)

- All testing complete
- Performance verified
- Documentation updated
- 0 regressions introduced

Week 3

- Second case study complete
- Case studies database ready (Week 4)

Weeks 4-8

- 3 case studies total
- Admin dashboard complete
- Testimonials integrated
- Contact forms working
- Government tender submitted
- Optimized & scaled

Emergency Recovery

If Chat Widget Breaks

1. Check error message (F12 → Console)
2. Verify: `ANTHROPIC_API_KEY` environment var set?
3. Verify: Database tables exist (Supabase Dashboard)
4. Verify: Model name correct (`claude-3-haiku-20240307`)
5. Check git diff (what changed?)
6. If unsure: `git checkout app/api/ai-assistant/route.ts`
7. Test: `npm run dev` → `/test-ai`

If TypeScript Breaks

1. `npm run build` (see error details)
2. Check `types/supabase.ts` (is it updated?)
3. Check for any type imports removed
4. Regenerate types if needed: `supabase gen types`
5. Fix imports in affected files
6. Verify: `npm run build`

If Accessibility Breaks

1. Run Lighthouse audit (F12 → Lighthouse)
2. Note new violations
3. Check git diff (what changed in layout/styles?)
4. Revert color/contrast changes if needed

5. Verify WCAG 2.1 AA again
6. Target: Return to 91-98/100 Lighthouse

Implementation Summary

Total Timeline: 8 weeks (adjust dates based on confirmation)

Week 1: ✅ COMPLETE (Accessibility + AI Assistant)

Weeks 2-8: Ready to execute with this corrected plan

Key Principles:

- Don't break what works - All current features are live ✅
- Preserve type safety - 100% TypeScript coverage ✅
- Maintain accessibility - WCAG 2.1 AA always ✅
- Test before commit - Local testing mandatory ✅
- Document changes - Clear commit messages ✅
- Verify deployment - Check live site after push ✅

Success = Delivering amazing features while protecting what already works.

Status: ✅ Week 1 Complete | Weeks 2-8 Ready to Start (with corrections)
Confidence: 95% (pending timeline confirmation)
Next Step: Confirm timeline → Begin Week 2 testing & validation