Phoenix Precision Agency - Comprehensive Project Context

1. Project Overview

Project Goals and Objectives

- **Primary Goal**: Create a modern, high-converting agency website for Phoenix small businesses
- **Value Proposition**: "NASA engineer brings aerospace precision to business websites"
- **Target Market**: Small businesses in Phoenix area needing website modernization
- **Core Message**: Transform outdated 2005-era websites into modern, performance-driven solutions

Key Architectural Decisions

- **Framework**: Next.js 15.4.5 with App Router (chosen for performance and SEO)
- **Styling**: Tailwind CSS v4 with custom design system
- **Type Safety**: Full TypeScript implementation for reliability
- **Route Groups**: Using `(marketing)` and `(dashboard)` for logical separation
- **Performance First**: Focus on Core Web Vitals and loading speed

Technology Stack

Core Dependencies:

- Next.js 15.4.5 (React 19.1.0)
- TypeScript 5.x
- Tailwind CSS 4.x
- Framer Motion 12.23.12 (animations)

UI Components:

- shadcn/ui components (Button, Card, Badge, Skeleton)
- Custom components built on Radix UI primitives
- Lucide React for icons

Backend Services:

- Upstash Redis (lead storage, rate limiting)
- Resend (email notifications)
- Clerk (admin authentication)

Testing:

- Jest for unit tests
- Playwright for E2E tests (multi-browser support)
- Testing Library for component tests

Team Conventions and Patterns

- **File Structure**: Feature-based organization with clear separation of concerns
- **Component Pattern**: Functional components with TypeScript interfaces
- **Styling**: Utility-first with Tailwind, custom CSS variables for theming
- **State Management**: React hooks and context for local state
- **Error Handling**: Centralized error boundaries and graceful fallbacks
- **Performance**: Lazy loading, code splitting, optimized images

2. Current State

Recently Implemented Features

1. **Interactive Demo Section**
    - Split-screen comparison (2005 vs modern site)
    - Auto-switching views with smooth zoom transitions
    - Progress indicators and animations
    - Responsive design with mobile optimizations
2. **Contact System**
    - Form with validation (react-hook-form)
    - Rate limiting via Upstash Redis
    - Email notifications through Resend
    - Lead storage in Redis
    - Success/error handling with user feedback
3. **Trust Building Elements**
    - NASA engineer badge with experience
    - Client testimonials section
    - 99.9% uptime guarantee
    - Performance metrics with animations
    - Portfolio showcase
4. **SEO & Performance**
    - Optimized meta tags
    - Open Graph implementation
    - Performance monitoring
    - Lazy loading for heavy components
    - Responsive images

Work in Progress

E2E Test Failures (189/280 failing):

Critical Issues:

1. **Duplicate Elements** - Multiple elements with same text causing test failures
2. **Missing Content Sections** - "Lightning-Fast Performance" section not found
3. **Touch Target Sizes** - Mobile buttons/links too small (< 48x48px)

4. **Responsive Issues** - Layout problems on mobile viewports

Current Todo List:

1. ✅ Create missing /contact and /portfolio pages - DONE (pages exist in `(marketing)` group)
2. 🔄 Fix duplicate elements causing strict mode violations (in_progress)
3. ⏳ Add missing content sections like Lightning-Fast Performance (pending)
4. ⏳ Fix touch target sizes and mobile test issues (pending)
5. ⏳ Run E2E tests to verify fixes (pending)

Known Issues and Technical Debt

1. **Test Infrastructure**
    - E2E tests need comprehensive update
    - Mock implementations need review
    - Test data needs standardization
2. **Content Consistency**
    - Contact info variations (some places show placeholder data)
    - Footer links need verification
    - Portfolio content needs real projects
3. **Performance Optimizations**
    - Bundle size analysis needed
    - Image optimization opportunities
    - Third-party script loading strategy

Performance Baselines

- **Lighthouse Scores** (target):
    - Performance: 95+
    - Accessibility: 100
    - Best Practices: 100
    - SEO: 100
- **Core Web Vitals**:
    - LCP: < 2.5s
    - FID: < 100ms
    - CLS: < 0.1

3. Design Decisions

Architectural Choices and Rationale

1. **App Router over Pages Router**
    - Better performance with React Server Components
    - Improved data fetching patterns
    - Built-in layouts and error handling

2. **Route Groups Strategy**
    - `(marketing)`: Public-facing pages with shared layout
    - `(dashboard)`: Admin area with authentication
    - Clear separation of concerns
3. **Component Architecture**
    - Atomic design principles
    - Reusable UI components via shadcn
    - Clear props interfaces with TypeScript

API Design Patterns

1. **Server Actions** for form submissions
2. **API Routes** for external integrations
3. **Edge Functions** for performance-critical paths
4. **Rate Limiting** at API level

Database Schema Decisions

Redis Data Structure:

```
leads:{email} - Contact form submissions
analytics:{metric}:{date} - Performance metrics
ratelimit:{ip} - Rate limiting counters
```

Security Implementations

1. **Input Validation** - Zod schemas for all user inputs
2. **Rate Limiting** - IP-based with Upstash
3. **CSRF Protection** - Built into Next.js
4. **Content Security Policy** - Strict CSP headers
5. **Authentication** - Clerk for admin access

File Structure

```
/app                    # Next.js App Router
  /(marketing)          # Public pages with shared layout
    /page.tsx           # Homepage
    /contact            # Contact page
    /portfolio          # Portfolio page
  /(dashboard)          # Protected admin area
    /dashboard          # Analytics dashboard
  /api                  # API routes
    /contact            # Contact form submission
/components             # Reusable React components
  /demo                 # Demo comparison components
  /forms                # Form components
```

```
  /ui               # Base UI components
/lib                # Utility functions and services
  /analytics.ts     # Analytics tracking
  /upstash.ts      # Redis integration
  /types.ts        # TypeScript definitions
```

4. Code Patterns

Coding Conventions Used

```
// Component Structure
interface ComponentProps {
  // Props with JSDoc comments
}

export function Component({ prop }: ComponentProps) {
  // Hook usage at top
  // Event handlers
  // Render logic
}

// Consistent naming
- Components: PascalCase
- Functions: camelCase
- Constants: UPPER_SNAKE_CASE
- Files: kebab-case or PascalCase for components
```

Common Patterns and Abstractions

1. **Loading States**: Skeleton components for better UX
2. **Error Boundaries**: Graceful error handling
3. **Lazy Loading**: Dynamic imports for heavy components
4. **Animation Patterns**: Framer Motion with consistent easing

Testing Strategies

1. **Unit Tests**: Critical business logic
2. **Integration Tests**: API routes and forms
3. **E2E Tests**: User journeys across browsers
4. **Visual Regression**: Screenshot comparisons

Error Handling Approaches

1. **Try-Catch Blocks**: Async operations
2. **Error Boundaries**: Component-level failures
3. **Fallback UI**: Loading and error states
4. **User Feedback**: Toast notifications

5. Agent Coordination History

Which Agents Worked on What

1. **Initial Setup Agent**: Project scaffolding, dependencies
2. **UI/UX Agent**: Design system, component library
3. **Backend Agent**: API routes, database integration
4. **Testing Agent**: Test suite setup, E2E tests
5. **Performance Agent**: Optimizations, monitoring
6. **Context Management Agent**: Documentation, knowledge transfer

Successful Agent Combinations

1. **UI + Backend**: Form implementations
2. **Testing + Performance**: E2E performance tests
3. **Context + All**: Knowledge sharing sessions

Agent-Specific Context and Findings

- **UI Agent**: Established Tailwind patterns, component library
- **Backend Agent**: Redis patterns, email integration
- **Testing Agent**: Playwright configuration, test patterns
- **Performance Agent**: Bundle analysis, optimization opportunities

Cross-Agent Dependencies

1. UI components → Backend APIs
2. Tests → All features
3. Performance → UI and Backend optimizations
4. Context → All agents for knowledge

6. Future Roadmap

Planned Features

1. **Analytics Dashboard**
    - Real-time visitor tracking
    - Conversion metrics
    - Performance monitoring
2. **Enhanced Portfolio**
    - Case studies with metrics
    - Before/after comparisons
    - Client testimonials
3. **Blog/Resources**
    - SEO-focused content
    - Technical guides

- Industry insights

4. **Client Portal**
    - Project status tracking
    - Document sharing
    - Communication hub

Identified Improvements

1. **Performance**
    - Implement service worker
    - Add resource hints
    - Optimize third-party scripts
2. **SEO**
    - Schema markup
    - XML sitemap
    - Robots.txt optimization
3. **Accessibility**
    - ARIA improvements
    - Keyboard navigation
    - Screen reader testing

Technical Debt to Address

1. **Code Quality**
    - Refactor duplicate code
    - Improve type coverage
    - Update dependencies
2. **Testing**
    - Increase test coverage
    - Add visual regression tests
    - Performance benchmarks
3. **Documentation**
    - API documentation
    - Component storybook
    - Deployment guides

Performance Optimization Opportunities

1. **Bundle Size**
    - Tree shaking improvements

    - Dynamic imports strategy
    - Dependency analysis

2. **Runtime Performance**
    - React component memoization
    - Virtual scrolling for lists
    - Web worker utilization

3. **Network Optimization**
    - HTTP/3 adoption
    - CDN strategy
    - Caching improvements

Deployment & Operations

Deployment Strategy

- **Platform**: Vercel (auto-deploy from GitHub)
- **Branch Strategy**:
    - `main`: Production
    - `develop`: Staging
    - Feature branches for development

Monitoring

- **Performance**: Vercel Analytics
- **Errors**: Built-in Next.js error reporting
- **Uptime**: External monitoring service

Environment Variables

```
UPSTASH_REDIS_REST_URL
UPSTASH_REDIS_REST_TOKEN
RESEND_API_KEY
CLERK_SECRET_KEY
NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY
CONTACT_EMAIL_TO (default: fmp321@gmail.com)
```

Quick Reference

Key Commands

```
pnpm dev          # Development server
pnpm build        # Production build
pnpm test         # Run tests
```

```
pnpm test:e2e     # E2E tests
git push          # Deploy via Vercel
```

Important Files

- `/app/(marketing)/*` - Public pages
- `/components/*` - Reusable components
- `/lib/*` - Utilities and helpers
- `/e2e/*` - E2E test suites
- `CLAUDE.md` - AI assistant instructions

Contact Information

- **Production Email**: contact@phoenixprecision.dev
- **Developer Email**: fmp321@gmail.com
- **Phone**: (602) 531-4111

GitHub Integration with E2E Tests

- **Workflow**: `.github/workflows/e2e-tests.yml`
- **Triggers**: Push to main/develop, pull requests
- **Test Matrix**:
  - Browsers: Chromium, Firefox, WebKit
  - Mobile: Pixel 5, iPhone 12
  - Runs in parallel for efficiency
- **Artifacts**: Test reports and screenshots on failure

Last Updated: 2025-08-03 **Version**: 1.0.0 **Status**: Active Development