Personal Website Specification Document

BRD • PRD • SDD • TSD

Author: Ray

2025-11-20 16:48:49

Version 2.0.0

Table of Contents

Chapter 1 — BRD (Business Requirements Document)

1. Project Background

This personal website showcases my professional identity as a Product Manager and Front-end Engineer. It highlights expertise in product strategy, user experience design, modern front-end engineering practices, design system creation, and an AI-assisted development workflow that demonstrates practical applications of contemporary AI tools for code generation, review, and developer efficiency.

2. Business Goals

- Strengthen personal brand and online presence for career growth and collaboration opportunities.

- Demonstrate competencies across PM, UX, and front-end engineering to hiring managers and collaborators.

- Present a transparent example of AI-assisted development processes, including toolchain, checks, and governance.

- Serve as a high-quality portfolio to attract full-time roles, contract work, speaking engagements, or collaborations.

3. Stakeholders

Primary Stakeholder:
- Ray (Owner, Developer, Product Lead)

Secondary Stakeholders:
- Recruiters and hiring managers
- Fellow developers and designers
- General visitors and potential collaborators
- Mentors and peers who may provide feedback or referrals

4. Scope

In-scope:

- Portfolio pages and project detail views
- Blog and article system supporting Markdown or JSON content
- Weather widget demonstrating data integration
- Notification system for site-level messages
- Responsive UI across multiple breakpoints with focus on mobile-first design
- A maintainable design system
- Performance optimizations and accessibility best practices

Out-of-scope:

- User accounts and authentication
- Backend CMS (content stored as static Markdown/JSON for now)
- Real-time collaboration features and comment systems
- E-commerce or transaction processing

5. Business Assumptions

- Majority of site visitors are passive browsers (recruiters, developers, designers).
- AI-assisted development tools used during construction will remain stable for the duration of the implementation.
- Hosting on Netlify or similar static hosting providers provides sufficient performance and CI/CD capabilities.

6. Risks

- Inconsistencies from AI-generated code requiring manual review and refactor.
- Performance degradation if media and scripts are not optimized.
- Maintaining the design system requires documentation and governance; drift may occur.
- Accumulating technical debt if scope expands without refactor cycles.

7. Business Success Metrics

- Portfolio views (monthly unique visitors to project pages)
- Recruiter contact frequency (number of inbound inquiries or interview requests)
- Blog readership and article completion rates
- System stability (uptime, error rate) and performance metrics (First Contentful Paint, Largest Contentful Paint)

Chapter 2 — PRD (Product Requirements Document)

1. Product Vision

Create a modern, AI-assisted, responsive personal website that clearly communicates my professional narrative, showcases high-quality case studies, and demonstrates practical AI-assisted front-end engineering workflows.

2. User Personas

Primary Personas:
- Recruiter: seeks concise evidence of skills, easy navigation to CV and projects.
- Designer/PM: inspects UX, flows, and design thinking.
- Developer: reviews code samples, architecture, and AI workflow details.
- Visitor: casual reader of articles and blog posts.

3. Core Features

- Portfolio/Works section: project cards, filters by tag/tech, detailed case studies with images, results, and links to code if available.
- Blog/Articles: list and article detail pages, reading progress, and semantic metadata for SEO.
- Weather widget: sample integration with 3rd-party API, graceful fallback states, caching.
- Notification system: UI banner / bell with dismissable messages (e.g., updated portfolio, talks).
- Responsive design with at least 6 breakpoints (XS → XL) and accessible navigation.
- Design system with tokens, components, and patterns (colors, spacing, typography).
- Performance optimizations: image optimization (WebP), code-splitting, lazy loading.

4. Non-functional Requirements

- Maintainability: modular ES6 structure, documented components, linting, and a minimal test suite.
- Performance: target initial load <2s on modern mobile networks and LCP <2.5s.
- Accessibility: WCAG AA baseline for color contrast, keyboard navigation, and semantic markup.
- Scalability: easily extendable to add a CMS or user-generated content later.

5. User Stories

- As a recruiter, I want to view technical project summaries and links so I can evaluate fit.
- As a visitor, I want to read articles with minimal distractions so I can focus on content.
- As a developer, I want to inspect the AI-assisted workflow so I can learn the process and tooling used.
- As the owner, I want an easy way to publish new articles or projects (via Git commits) without complex backend requirements.

6. Success Metrics

- Bounce rate for project pages should be below industry baseline for portfolios.
- Article read-through rate (percentage of visitors who reach the end of an article).
- Site load performance and search visibility (Core Web Vitals).

Chapter 3 — SDD (Software Design Document)

1. System Overview

A modular ES6 front-end application deployed as a static site (Netlify). The system uses a component-based structure with reusable UI primitives, a design token system, and light-weight services for data fetching (e.g., weather API, article loader).

2. Architecture

Layered Architecture:
- UI Layer: HTML, CSS (modern features like CSS variables), and vanilla ES6 JavaScript or a lightweight framework if desired (React/Vue/Svelte).
- Components: isolated units (ProjectCard, ArticleList, WeatherWidget, NotificationPanel).
- Services: small modules for network calls, caching, and error handling (weatherService, articleService).
- Utilities: animation helpers, formatters, lazy-load handlers.

3. Design System

Design language and tokens:
- Visual style: glassmorphism with backdrop-filter blur, warm earth-tone palette.
- Tokens: color, spacing, radius, typography size scale, elevation tokens.
- Component rules: accessibility attributes, responsive behavior, states (hover, focus, active).
- Documentation: a living style guide page with component examples and usage guidelines.

4. Component Design

Project Card:

- Fields: title, short description, tech tags, hero image, link to detailed case study.
- Interactions: hover reveal of short excerpt, lazy load images, accessible focus states.

Article Module:

- List view and detail view; support for reading time, table of contents, and semantic headings.
- Markdown rendering pipeline with code block highlighting and image optimization.

Weather Widget:

- Shows temperature, condition, icon, and location. Caches data for X minutes and gracefully falls back to cached/default state on errors.

Notification Panel:

- Supports priority levels, timestamps, dismiss action, and optional persistence (localStorage) for dismissed messages.

5. Performance Design

- Use IntersectionObserver for lazy loading images and components.
- Serve images as WebP with appropriate srcset for responsive sizes.
- Limit large layout-shifting elements and use transform/opacity for animations.
- Minify and bundle assets; prefer HTTP/2 compatible hosting and CDN.

6. Extensibility & Maintainability

- Clear module boundaries with a component-first folder structure.
- Version controlled via Git with branches for features.
- Document AI-generated code review guidelines, including manual review checklist and tests.

Chapter 4 — TSD (Technical Specification Document)

1. Technology Stack

- Front-end: HTML5, CSS3, ES6 JavaScript. Optional lightweight framework (React recommended for component lifecycle, but vanilla JS is acceptable for a small site).
- Build: Vite or comparable bundler for fast dev builds; Netlify for deployment and CI.
- AI Tools: GitHub Copilot, Claude Sonnet 4 (used for ideation, code snippets, and documentation drafts).
- Tooling: ESLint, Prettier, Husky for pre-commit hooks, and a minimal testing library (Jest or Vitest) for utilities.

2. Module Specifications

Weather Module — weatherService.js:

- Function: fetchWeather(location)
- Inputs: location identifier (city name or coordinates)
- Outputs: { temperature, humidity, icon, status, timestamp }
- Error handling: return { status: 'unavailable', icon: 'default', message: 'Unable to fetch' } and use cached data if available.

Blog Module — articleService.js:

- Methods: getAllArticles(), getArticleBySlug(slug)
- Data: stored as Markdown or JSON files in /content. Front-matter includes title, date, tags, summary, hero image.
- Caching & indexing: pre-generate article index at build time for performance.

Notification Module — notificationService.js:

- Data model: { id, priority: ['low','normal','high'], message, time, persistent }
- Methods: addNotification(item), removeNotification(id), renderNotifications().
- Persistence: localStorage optional for persisted dismissals.

3. Layout & Components

- Core files: navbar.js, footer.js, projectCard.js, articleList.js, weatherWidget.js, notificationPanel.js
- CSS: root variables, responsive breakpoints, utility classes, component styles.
- Accessibility: semantic HTML, ARIA labels for interactive widgets, skip-to-content link.

4. Configuration

- Netlify: build command (npm run build), publish directory (dist or build), environment variables for any API keys.
- Asset directory: /public/assets/images, /public/icons; naming conventions and fallback rules.
- Image fallback: provide default hero and icon images when content is missing.

5. Testing

- Unit tests for utility functions (date formatters, fetch wrappers).
- Visual regression checks (manual snapshots or lightweight tools) for key components.
- Error handling tests for widgets (simulate API failures and check UI fallback).

6. Deployment Process

- Git flow: feature branches, PR reviews, main branch deploys automatically to Netlify.
- Versioning: use semantic tags for releases (v0.1.0).
- Rollback: Netlify provides deploy history for rollbacks; document quick rollback steps.

Appendix A — Acceptance Criteria & KPIs

- All core pages render without errors and meet WCAG AA contrast levels.
- Core user flows (view project, view article) complete without JS console errors.
- Performance: FCP <1.5s on desktop, LCP <2.5s target; Lighthouse score >90 (desktop target).
- Monitoring: errors tracked in Sentry or log collector; uptime >99.5%.

Appendix B — Implementation Roadmap (Milestones)

Phase 0 — Discovery & Design (1–2 weeks): finalize content, wireframes, and design tokens.
Phase 1 — Core Components & Layout (2–3 weeks): build ProjectCard, Article module, Navbar, Footer.
Phase 2 — Widgets & Integrations (1–2 weeks): Weather widget, Notification system, SEO and metadata.
Phase 3 — Performance & Accessibility (1 week): optimize images, run audits and fix issues.
Phase 4 — Launch & Monitoring (1 week): deploy, configure analytics, and set up monitoring.

Appendix C — Risk & Mitigation Plan

- Risk: AI-generated bugs — Mitigation: mandatory code review, unit tests, and linting.
- Risk: Image bloat — Mitigation: automated image optimization and strict size limits.
- Risk: Design drift — Mitigation: documented design tokens and component examples.

Appendix D — Sample API Schemas & Data Models

Article front-matter schema (example):

```
{
  'title': 'Project Title',
  'date': 'YYYY-MM-DD',
  'slug': 'project-title',
  'tags': ['frontend','design'],
  'summary': 'Short summary',
  'hero': '/assets/images/hero.webp'
}
```

Weather response schema (example):

```
{
  'location': 'Taipei',
  'temperature': 25.3,
  'humidity': 78,
  'icon': 'partly-cloudy',
  'status': 'ok',
  'timestamp': '2025-11-21T00:00:00+08:00'
}
```

Final Notes

This document expands the original outline into a professional specification suitable for sharing with collaborators or including in a hiring portfolio. It is designed to be actionable with clear acceptance criteria, test considerations, and a phased roadmap.

If you want, I can generate additional artifacts from this document: a one-page executive summary, a visual roadmap for the website, sample component code (ProjectCard, WeatherWidget), or a ready-to-deploy Netlify configuration file.

— End of Document —