# SprintPilot — Project Documentation

This is a comprehensive, presentation-ready project document for SprintPilot. Use this as the base content for generating a PDF for presentations, handouts, or investor/demo decks.

---

## 1. Project Overview

Project name: SprintPilot

Short description:
SprintPilot is an all-in-one agile product management workspace that helps product teams plan, prioritize, and deliver products faster. It combines backlog management, roadmaps, PRD creation, wireframing, competitive analysis, and AI-powered assistance in a modern, delightful interface.

Primary audience: Product managers, UX designers, engineering leads, startup founders.

Key value propositions:

- A single workspace that covers strategy (roadmaps), execution (backlog/tasks), and discovery (personas/competitive analysis).
- AI-powered helpers for faster PRD generation and content suggestions.
- Modern, responsive UI with modular components for quick iteration.

## 2. Quick Facts & Tech Stack

- Frontend: React 18 + TypeScript
- Build: Vite
- Styling: Tailwind CSS
- UI primitives: Radix / shadcn components
- Data & Auth: Supabase
- AI: Google Gemini via @google/genai
- State & network: React Query
- Drag & Drop: @dnd-kit
- Charts: Recharts
- PDF / export utilities: jspdf / html2canvas

Important scripts (from package.json):

- npm run dev — start dev server
- npm run build — production build
- npm run preview — preview build

## 3. Project Structure (high level)

src/

- App.tsx — app root / routing
- main.tsx — entry
- index.css — project styles
- components/ — UI and feature components

    - landing/ — public landing pages (hero, features, faq, etc.)
    - dashboard/ — dashboard layout
    - ui/ — design system building blocks (button, input, dialog, toast, cards, etc.)

- pages/ — route-based page components including dashboard pages: backlog, roadmaps, persona, wireframes, prd-builder, settings
- lib/ — API clients and helpers (supabase.ts, jiraApi.ts, geiminiApi.ts, utils.ts, storage.ts)
- context/ — AuthContext.tsx for auth state
- hooks/ — shared hooks

Database definitions (supabase schema):

- 001_create_personas.sql
- 002_create_prd_tables.sql
- 003_create_workspaces.sql
- 004_create_tasks.sql
- 005_create_api_keys.sql
- 006_add_prd_content_column.sql
- 007_create_wireframes.sql
- 008_create_competitors.sql
- 009_create_roadmaps.sql

## 4. Features & Pages (detailed)

Landing

- Hero, feature section, CTA — marketing material to convert users.
- FAQ, footer and design system showcasing.

Authentication

- Signup, login, forgot/reset password pages and AuthContext for session handling.
- Integration with Supabase Auth for email/password and access control.

Dashboard & Workspace

- Multi-workspace support (see 003_create_workspaces.sql).
- A central Dashboard with cards, charts, and activity feed.

Backlog

- Issue/task CRUD (see 004_create_tasks.sql).
- Prioritization controls, sorting, filters.
- Drag-and-drop reordering and Kanban-style status columns using @dnd-kit.

Roadmaps

- Visual roadmap creation with drag-and-drop (see 009_create_roadmaps.sql).
- Assign releases, owners, and link to backlog items.

PRD Builder

- Structured PRD templates (see 002_create_prd_tables.sql, 006_add_prd_content_column.sql).
- Export to PDF using jspdf and html2canvas for snapshots.

Personas & Competitive Analysis

- Create and view personas (001_create_personas.sql).
- Competitive tracking (008_create_competitors.sql) with fields for strengths/gaps.

Wireframing

- Simple inline wireframing editor and storage (007_create_wireframes.sql).

AI Assistants

- Gemini integration for content generation and suggestions (lib/geiminiApi.ts or lib/ai-persona.ts).
- Examples: PRD content generation, persona synthesis, roadmap recommendations.

## 5. Data Model & Important Database Tables (summary)

- personas: user personas with demographic and behavioral attributes. (001)
- prd_tables / prd_content: PRD metadata and content storage (002,006)
- workspaces: multi-tenant workspace objects with members and roles (003)
- tasks: backlog items, statuses, priorities, estimates, linked PRDs (004)
- api_keys: stored API keys for integrations (005)
- wireframes: simple wireframe JSON blobs and versions (007)
- competitors: competitor records for analysis (008)
- roadmaps: roadmap lanes, items, and milestones (009)

Note: See SQL files in /supabase-schema for the exact columns, constraints, and sample seeds.

## 6. Environment Variables & Secrets

Typical .env values (prefixed with VITE_ as used in client):

- VITE_SUPABASE_URL
- VITE_SUPABASE_ANON_KEY
- VITE_GEMINI_API_KEY (or whichever key/provider is used)
- Optional: VITE_JIRA_API_KEY or other third-party integrations

Security notes:

- Never commit .env to source control. Use environment or secret manager in production (Vercel, Netlify, or cloud provider secrets).
- For server-side secrets like Supabase service role keys, store them server-side only.

## 7. How to Run Locally (step-by-step)

1. Clone repository
2. Install dependencies
    - npm install
3. Create .env with required keys (see section above)
4. Start dev server
    - npm run dev
5. Open http://localhost:5174

Optional: Run npm run build and npm run preview to test production build locally.

## 8. Testing & Linting

- Linting: npm run lint (ESLint)
- Unit/UI tests: none included by default; recommended: Jest + React Testing Library or Vitest.

Suggested tests to add:

- Auth flows (login/signup/reset)
- PRD builder content save/load
- API client mocks for Supabase and Gemini
- Component snapshot tests for critical UI atoms

## 9. Deployment

Recommended deployment platforms:

- Vercel or Netlify for frontend (Vite) with environment variables set in the provider.
- Supabase for database & Auth.

Deployment checklist:

- Set production Supabase keys in platform secret store
- Ensure CORS / redirect URIs are set in Supabase for your domain
- Set a CI build step to run npm ci and npm run build then publish

## 10. Security & Privacy Considerations

- GDPR/Privacy: if storing user data, add privacy policy and allow data export/removal.
- Rate limit/monitor third-party AI API usage and ensure API keys are rotated.
- Validate and sanitize any uploaded wireframe content before rendering.

## 11. Performance & Scalability Notes

- Client-heavy SPA; push heavy compute (large reports, AI interactions) to serverless functions when possible.
- Use pagination for large backlogs and server-side filtering.
- Cache repeated reads (user persona lists, roadmap items) with React Query and short TTL.

## 12. Roadmap & Next Steps (suggested)

Short term (0-3 months):

- Add authentication guards and role management UI
- Add unit and integration tests for critical flows
- Improve PRD export (templated DOCX / PDF with rich formatting)

Medium term (3-6 months):

- Collaboration features: comments, mentions, real-time cursors (WebSockets)
- Import/export from Jira/Trello

- Advanced AI features: auto-prioritization and insights

Long term (6-12 months):

- Multi-organization billing and admin panels
- Plugin architecture for integrations
- Native mobile apps or responsive offline mode

## 13. Appendix

A. Useful files & locations

- src/lib/supabase.ts — Supabase client
- src/lib/geiminiApi.ts — Gemini AI wrapper (or ai-persona.ts)
- src/pages/prd-builder.tsx — PRD builder UI
- supabase-schema/ — SQL schema files

B. Commands cheat-sheet

- npm install — install deps
- npm run dev — dev server
- npm run build — production build
- npm run preview — preview build
- npm run lint — linting

C. How to convert this doc to PDF (options)

- Using pandoc (if installed):
    - pandoc PROJECT_DOCUMENTATION.md -o SprintPilot_Project.pdf --pdf-engine=xelatex
- Using Node (npx markdown-pdf):
    - npx markdown-pdf PROJECT_DOCUMENTATION.md -o SprintPilot_Project.pdf
- Using a browser: open PROJECT_DOCUMENTATION.md in an editor that previews markdown (VS Code) and use Print -> Save as PDF.