Product Manager Intern Assignment

Daniel Freire Mendes

February 1, 2026

1. Project Overview

Project Links

- **Live Prototype:** https://cfchallenge.pages.dev
- **GitHub Repository:** https://github.com/danielfmendes/cf_challenge

Product Interfaces

- **Dashboard:** A visual command center that summarizes feedback totals, sentiment trends, and high-priority issues to eliminate manual data review.
- **Ingestion Playground:** A simulation area to send data to the API using pre-set templates. Users can modify the JSON text content as long as the original key structure remains unchanged.
- **Error Detail Page:** A focused deep-dive for individual problems that provides an AI-generated root cause analysis and a suggested fix to help teams resolve issues faster.

2. Product Insights

Insight: Recovery Friction for Leaked Database IDs

Problem: After accidentally committing a D1 UUID to a public repository, I found no native way to rotate the ID. I had to manually create a new database and migrate data via `wrangler d1 export` and `execute`, a tedious process that stalled development momentum.

Suggestion: Cloudflare should provide a "Rotate ID" button in the dashboard to regenerate UUIDs instantly. Additionally, supporting "Alias Bindings" would allow `wrangler.toml` to reference pointer names while keeping actual UUIDs in secure environment variables.

Insight: Inconsistent Git Integration (Pages vs. Workers)

Problem: Coming from Vercel and Railway, I expected a "Connect to Git" experience for the entire platform. While Cloudflare Pages offers native Git integration, Cloudflare Workers does not. To deploy my Worker automatically, I had to manually configure a GitHub Actions workflow, creating a disjointed setup where the frontend and backend are deployed via different mechanisms.

Suggestion: Bring the "Pages Experience" to Workers. Allow users to connect a GitHub repository directly to a Worker service in the dashboard, enabling automatic deployments on push without needing to write custom YAML CI/CD pipelines.

Insight: Misleading DNS Status Indicators

Problem: When I connected my custom domain (`danielfreiremendes.com`), the dashboard immediately reported it as "Managed by Cloudflare." However, the site was inaccessible for nearly two days. I initially thought I had misconfigured the Worker routes, wasting time debugging code when the issue was actually standard DNS propagation.

Suggestion: Improve the status state. Instead of a green "Active" checkmark immediately, display a "Propagating" status with a warning: *"DNS changes can take 24-48 hours. Your site may not be visible yet."* Ideally, include a "Test Propagation" button that checks the record availability in real-time.

Insight: Information Architecture of "Workers & Pages"

Problem: I was initially confused by the platform's decision to co-locate these services. The unified dashboard conflates two fundamentally different resource types: *Hosting* (Pages) and *Compute* (Workers). This lack of separation forces users to rely solely on naming conventions (e.g., naming a worker `api-xyz`) to distinguish frontend projects from backend services, increasing cognitive load during navigation.

Suggestion: Align the UI with the developer's mental model by introducing resource categorization. Grouping items by intent (e.g., "Web Projects" vs. "Serverless Functions") or adding distinct "Type" columns would allow users to scan for the *kind* of resource rather than just the name.

3. Architecture Overview

3.1 The Cloudflare Stack

I selected the following services to handle the end-to-end data lifecycle:

- **Compute (Workers & Workflows):** A Worker acts as the API entry point, which then triggers a **Cloudflare Workflow** to handle long-running AI enrichment reliably with automatic retries.
- **Intelligence (Workers AI):** Utilized for real-time sentiment analysis, urgency scoring, and technical summarization of raw user feedback.
- **Storage (D1 & Vectorize):** Relational metadata is stored in **D1 (SQL)**, while **Vectorize** stores semantic embeddings to group similar user issues automatically.
- **Frontend (Pages):** A Vite + Tailwind dashboard is hosted on **Cloudflare Pages**, providing a fast, global interface for data visualization.

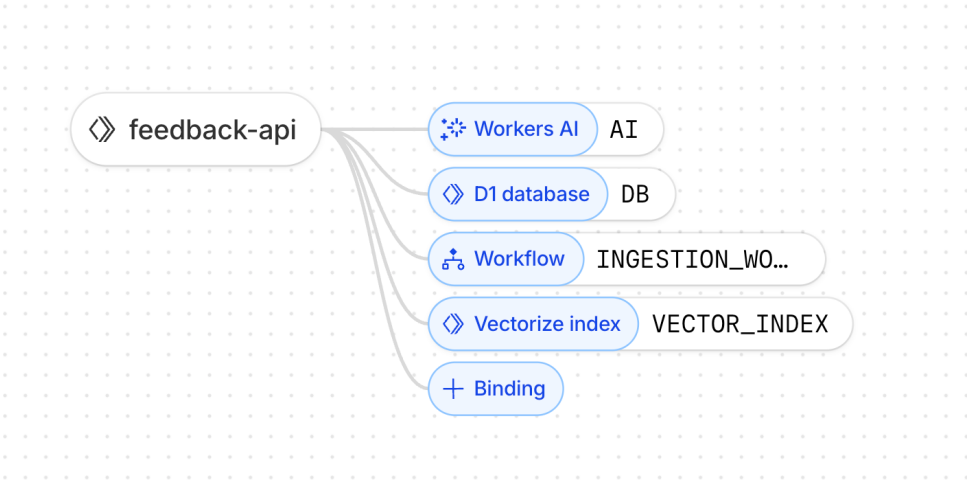

Figure 3.1: Cloudflare Worker Bindings: Integrating AI, SQL, and Workflows.

3.2 Reliability Pivot: Why Workflows?

Initially, high-volume AI requests caused standard Worker timeouts. I pivoted to **Cloudflare Workflows** to decouple ingestion from processing. This enables automatic retries of AI processing steps when models are rate-limited, ensuring reliable enrichment and persistence with no dropped events.

3.3 CI/CD Pipeline

The stack is deployed via **GitHub Actions**. Sensitive IDs are managed as GitHub Secrets to maintain a secure, public repository while ensuring the backend and frontend stay synchronized on every push to `main`.

4. Vibe-Coding Context

To accelerate prototyping, I combined Cloudflare Documentation for architectural validation with Gemini 3 Pro for implementation logic. I manually verified service bindings before prompting the AI to generate the Vite + shadcn/ui frontend and Worker API boilerplate.

Representative Prompts:

- *"Build a Cloudflare Worker POST endpoint that triggers an asynchronous workflow with a UUID, returning the ID as JSON."*
- *"Create a responsive React dashboard with shadcn/ui displaying three metric cards (Total Feedback, Critical Issues, Sentiment)."*