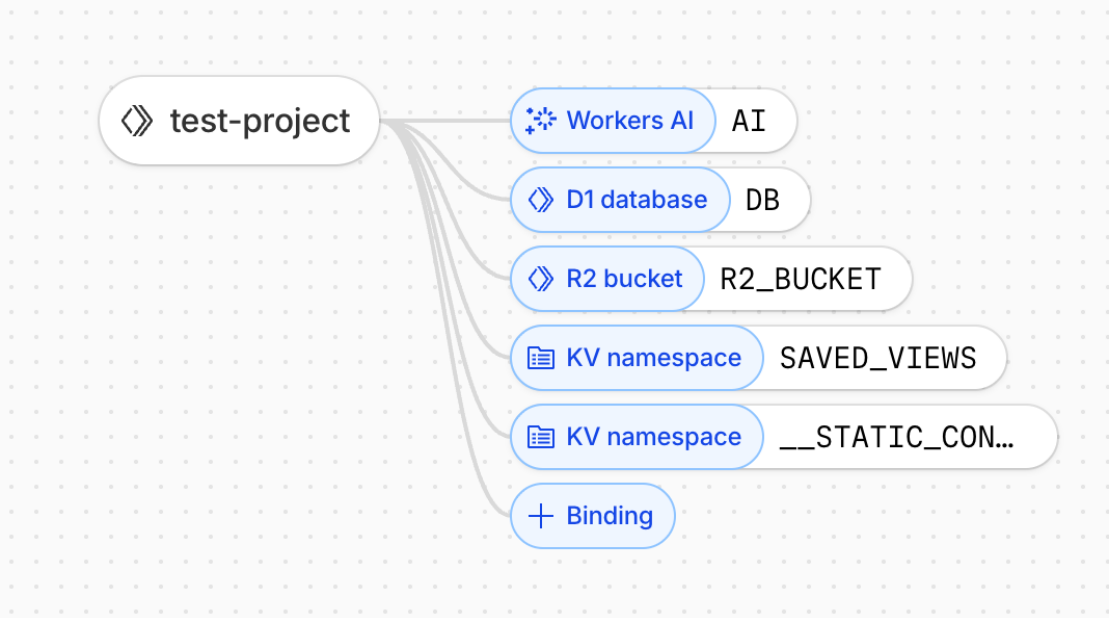
Product Manager Intern Assignment

Part 1: The Build Challenge

- **Project Links**
  - Cloudflare Workers: test-project.hong10054010.workers.dev
  - GitHub repository: https://github.com/hong10054010-art/test-project

- **Architecture Overview**



1 Cloudflare Products Used

1.1 Cloudflare Workers (Core Runtime)

Why: Workers serves as the backbone of the application, providing serverless execution for all API endpoints and static file serving. It enables **unified platform** that all other Cloudflare products integrate seamlessly through Workers bindings

Usage in this project:

- Main entry point (src/index.js) handles routing for API endpoints and serves the dashboard
- API endpoints in functions/api/ process requests and interact with other Cloudflare services
- Static HTML content is embedded directly in the Worker for optimal performance

1.2 Workers AI (AI Analysis Engine)

Why: Workers AI provides on-demand AI inference without managing infrastructure. It's used to:

- **Extract themes** from unstructured feedback text
- **Analyze sentiment** (positive, neutral, negative)
- **Determine urgency** levels (low, medium, high, critical)
- **Assess value** of feedback (low, medium, high)
- **Generate summaries** and extract keywords
- **Provide recommendations** based on aggregated data

Usage in this project:

- functions/api/process.js: Processes raw feedback using @cf/meta/llama-3.1-8b-instruct model to enrich data
- functions/api/ai-advice.js: Generates actionable recommendations based on filtered feedback data
- All AI processing happens at the edge, ensuring fast response times

1.3 D1 Database (Primary Data Storage)

Why: Serverless SQL database that provides:

- **Structured data storage**: Perfect for storing feedback metadata and AI analysis results
- **SQL interface**: Familiar query language for complex filtering and aggregation
- **Zero configuration**: No database servers to manage

Usage in this project:

- raw_feedback table: Stores original feedback from various sources (support tickets, GitHub, Discord, email, Twitter)

- enriched_feedback table: Stores AI-processed insights (themes, sentiment, urgency, value, summaries, keywords)
- functions/api/query.js: Performs complex SQL queries with filters for product, platform, country, and time range
- Indexes optimize query performance for dashboard visualizations

1.4 R2 Storage (Data Backup & Archive)

Why: R2 is included to demonstrate how raw feedback could be archived or reprocessed in a production scenario. In this prototype, it represents a scalable data durability layer rather than a fully exercised backup system.

Usage in this project:

- functions/api/process.js: After processing feedback with AI, raw data is backed up to R2 bucket
- Ensures data durability and enables future data recovery if needed

1.5 KV Namespace (User Preferences Storage)

Why: KV is a key-value store ideal for:

- **Fast lookups**: Sub-millisecond read latency for user preferences
- **Simple data model**: Perfect for storing saved filter configurations

Usage in this project:

- functions/api/save-view.js: Stores user-defined filter combinations (product, platform, country, time range) with custom names
- Enables users to save and quickly access frequently used dashboard views
- Lightweight storage for user preferences without database overhead

2 Data Flow

Some components are implemented as functional prototypes using mock data to demonstrate product intent and system design, rather than production-ready completeness.

- **Ingestion**: Mock feedback data is generated and stored in D1 raw_feedback table
- **Processing**: Workers AI analyzes raw feedback to extract themes, sentiment, urgency, and value
- **Enrichment**: Processed insights are stored in D1 enriched_feedback table
- **Backup**: Raw feedback is backed up to R2 for durability
- **Query**: Dashboard queries D1 with filters to display visualizations
- **Recommendations**: Workers AI generates actionable insights based on aggregated data
- **Persistence**: User filter preferences are saved to KV for quick access

3 Performance Optimizations

- **Indexed queries**: D1 indexes on frequently filtered columns (source, product_area, country, created_at)
- **Embedded HTML**: Static dashboard HTML embedded in Worker for instant delivery
- **Edge caching**: Cloudflare's global network caches responses automatically
- **Batch processing**: Feedback processed in batches for efficient AI inference

- ## Vibe-coding Context
  - **Vibe coding platform**: Cursor
  - **Initial prompts:**

    The current UI focuses on validating data flow and interaction patterns. Visual polish and advanced charting are intentionally lightweight for this prototype.

    Role Definition

    You are a **Product Manager** who supports companies in **data-driven decision-making and product planning**.
    You specialize in transforming **large, complex datasets** into **clear, intuitive, and visually compelling web experiences** that enable smooth communication and insight discovery.

    Objective

    Design and prototype a **User Feedback Aggregation & Analysis Tool** using **Cloudflare's product ecosystem**.
    The tool processes high-volume, unstructured feedback from multiple channels—such as **support tickets, GitHub, community platforms, email, and Twitter/X**—and helps Product Managers extract:

    - **Themes**
    - **Urgency**
    - **Business Value**
    - **Sentiment / Emotional Tone**

    Design Style

    - Reference the **attached UI visualization example** for layout and presentation.
    - Adapt the color palette to **Cloudflare's orange-based brand colors**.
    - The interface should feel **clean, modern, and data-driven**, suitable for professional PM workflows.

Content & Interaction Logic

1. Filter Controls

Dropdown filters allowing users to select: Product, Time range, Country and Platform / Channel (e.g., support, GitHub, social)

2. Data Visualizations

Present insights in clearly separated sections, including breakdowns by:

- Theme
- Platform
- Country
- Product
- Time
- Total feedback volume

Use appropriate chart types based on the data, such as:

- Pie charts
- Bar charts
- Line charts
- Geographic maps

3. Save & Share

Include **Save** and **Share** actions that allow Product Managers to:

- Save the current filtered view
- Share the visualized insights and data with others via a link

4. AI Recommendations

Provide an optional **"AI Recommendations"** button that, when activated, generates forward-looking insights and suggested actions based on the analyzed feedback.

5. Layout Hierarchy (Top to Bottom)

1. Main title: **"Feedback Insights"**
2. Filter controls
3. Total feedback count
4. Individual visualization sections and statistical breakdowns

Technical Implementation

- Implement the prototype based on the **attached system flow diagram**.
- Use **Cloudflare products** (e.g., Pages, Workers, D1, Workers AI, KV) to support data ingestion, processing, analysis, and presentation.

Part 2: Cloudflare Product Insights

◆ Insight 1

- **Title:** Unclear mental model when choosing between Workers and Pages

- **Problem:** When starting a new project on Cloudflare, users are often required to choose between Workers, Pages, and Pages Functions early in the setup process.

  In practice, users may restart or migrate projects unnecessarily after realizing the initial choice was suboptimal, or avoid experimenting altogether due to fear of "choosing the wrong option.

  While the documentation explains each product individually, users must mentally piece together the differences and trade-offs across multiple pages. As a result, the decision often feels premature and high-risk, particularly for users who are still exploring their use case rather than executing a well-defined technical plan.

  This ambiguity increases cognitive load at the very beginning of the user journey, where clarity and confidence are most critical.

- **Suggestion:** This friction can be reduced by providing clearer, use-case-driven guidance at the moment of product selection:

  - A lightweight decision wizard that asks about the user's goal (e.g. static site, API, data-heavy application, AI-powered workflow) and recommends an appropriate starting point
  - Clear "When to use / When not to use" comparisons directly in the project creation flow

  By reframing the decision around user intent rather than product taxonomy, Cloudflare can help users move from exploration to execution with greater confidence, faster time-to-value, and less cognitive overhead.

◆ Insight 2

- **Title:** Inconsistent bindings visibility between Pages and Workers

- **Problem:** When using Cloudflare Workers, the dashboard provides a clear and helpful bindings visualization that shows which resources (such as D1, R2, KV, and Workers AI) are connected to a Worker. This visual representation reinforces the mental model that Cloudflare projects expose their dependencies in a transparent and inspectable way.

  However, when building similar functionality using Pages, this bindings visualization is not available. From a user's perspective, the underlying capabilities are closely related, yet the dashboard experience differs significantly.

  This inconsistency creates a gap between user expectations and actual product behavior at a critical point in the workflow, after users have already committed to building something.

  For PMs and non-engineering users, this friction is especially disruptive, as they rely heavily on visual cues in the dashboard to build confidence and understand system structure without diving into code.

- **Suggestion:** Cloudflare could reduce this confusion by setting clearer expectations and improving consistency around bindings visibility.
  - Providing a simplified bindings summary for Pages projects, even if it is read-only or less detailed than the Workers visualization
  - Explicitly indicating in the Pages dashboard that bindings visualization is not available, along with a short explanation of why
  - Offering a clear comparison or tooltip that explains differences in observability between Workers and Pages

◆ Insight 3

- **Title:** Cloudflare resources lack reverse usage visibility
- **Problem:** When working with Cloudflare Workers, bindings provide a clear and intuitive view of which resources (such as D1, R2, KV, and Workers AI) are connected to a specific Worker.

  However, when navigating to the D1 or R2 dashboards, there is no equivalent way to see which Workers or projects are consuming that resource. The visibility remains one-directional: users can see what a Worker uses, but not where a database or bucket is being used. This lack of reverse visibility makes it difficult for users to make safe and confident decisions about resource management.

  For example, it becomes hard to answer simple but critical questions such as: “Can I safely delete this database?”

  Without knowing which projects depend on a given D1 database or R2 bucket, users risk either accidentally breaking active projects or avoiding cleanup altogether due to uncertainty, leading to unused or orphaned resources. As projects scale or multiple Workers share the same resources, this friction increases operational risk and dependency confusion across teams.
- **Suggestion:** Adding a simple “Used by” panel on resource dashboards would significantly improve clarity around ownership and dependency management, especially for PMs and non-engineering users. Even a read-only view listing consuming Workers or projects would be sufficient to support safer decision-making.