CASE STUDY

A user-facing application enabling low-cost service updates and full client ownership

Decoupled Service Architecture & Workflow Orchestration

The client required a user-facing service that enabled users to access, interact with, and act on data drawn from multiple external sources. The platform needed to transform inconsistent incoming data into a structured and reliable format, presenting it through a clear, simple, and accessible interface.

As the service expanded, new data sources, features, and user requirements were introduced regularly. Existing approaches made even small updates costly and difficult to manage, limiting the client's ability to evolve the platform efficiently and respond to user needs.

ENVIRONMENT	REQUIREMENT	ARCHITECTURE	OUTCOME
Public-facing, data-driven digital service	**Low-cost updates without service disruption**	**Decoupled frontend, logic, and integration layers**	**Maintainable system with full client ownership**

The challenge

The client required a system that could deliver a high-quality user experience while remaining flexible enough to support ongoing change. As the platform grew, tightly coupled components and inconsistent data handling made updates increasingly complex and costly.

- Presenting complex, multi-source data through a clear and accessible interface.
- Integrating external APIs with inconsistent formats and reliability.
- Introducing new features and updates without impacting existing functionality.
- Avoiding dependency on proprietary systems or opaque logic.

Without a modular structure, the platform risked becoming difficult to maintain, increasing development costs and raising the likelihood of user-facing disruption during updates.

Our approach

We designed and built a user-facing web application supported by a modular backend architecture, allowing each part of the system to evolve independently. Responsibilities were clearly separated to ensure changes could be made safely without impacting the overall service.

- A modern, responsive frontend focused on clarity, accessibility, and usability.
- A decoupled backend layer responsible for data processing and orchestration.
- Integration with multiple external APIs, with validation and normalisation of incoming data.
- Clear separation between presentation, business logic, and data handling.
- Structured interfaces allowing new features and data sources to be added with minimal impact.

The system was designed to avoid black-box behaviour, with transparent data flows and clear documentation, ensuring the client retained full visibility and control over the platform.

Results

The resulting application provided a stable and user-friendly interface, supported by a backend designed for adaptability and long-term use.

- A clear and accessible user experience for interacting with complex data.
- Reliable integration across multiple external data sources.
- Low-cost introduction of new features and updates.
- Reduced risk of disruption when evolving the service.

The platform could be extended and improved without large redevelopment cycles, reducing both cost and operational risk.

What this enabled

By introducing a modular and transparent system design, the service moved from a rigid, high-maintenance structure to a flexible and sustainable platform.

- Faster iteration and improvement based on user needs.
- Lower long-term operational and development costs.
- Full client ownership of code, data, and system behaviour.
- Long-term maintainability without vendor lock-in.

Technical environment (representative)

- Modern web application framework (component-based frontend)
- API-driven backend architecture
- Integration with external data services
- Structured data transformation and validation layers
- Containerised deployment and cloud infrastructure