CASE STUDIES

ArcadeProfile

By Austen Tucker

Why I built it

I needed more than a portfolio page. Fiction, essays, product work, subscriptions, and serialized releases all have different shapes, but visitors should encounter one coherent, readable public site.

ArcadeProfile is that product and its publishing machinery. It gives readers full semantic web pages while giving me a content model, scheduling queue, delivery records, and authoring tools behind the public surface.

Build snapshot

- Public product: Next.js and React pages for stories, projects, products, feeds, and subscriptions
- Content platform: Payload CMS, PostgreSQL, structured collections, previews, and media storage
- Delivery: ActiveCampaign audience state with Postmark transactional and newsletter delivery
- Operations: Scheduled publishing, delivery audit records, scoped MCP tools, and path revalidation

What I did

I made the web page the primary publication

Stories and essays render as semantic HTML with real headings, navigation, reading order, and RSS discovery. Reading controls change type, contrast, line height, and measure without turning the work into a separate document-only experience.

I built publishing as a stateful workflow

A post moves through draft, scheduled, published, and newsletter-sent states. A scheduled job promotes due work, resolves its audience, sends through Postmark, and records the attempt before the item is considered sent. Validation blocks impossible dates and the delivery trail distinguishes acceptance from confirmed delivery.

I separated audience ownership from mail delivery

ActiveCampaign owns contacts, consent, subscription lists, and segmentation. Postmark owns messages that leave the application. Keeping those responsibilities explicit avoids two systems quietly competing to define what “subscribed” or “sent” means.

I added bounded authoring interfaces

The Payload admin supports structured editing and previews. The same content system exposes scoped MCP tools over local and HTTP transports, with separate read and write credentials. New tools share one implementation instead of drifting between entry points.

What I learned

Publishing is a state machine

A date field and a cron job are not enough. Publication, audience resolution, delivery acceptance, confirmation, retry safety, and rollback each need a recorded state. The model became clearer once those transitions were named.

“Sent” needs evidence

An API accepting a batch does not prove that messages reached recipients. Message identifiers, webhook events, delivery counts, and reconciliation make the claim inspectable without pretending that every downstream outcome is known immediately.

Accessibility belongs in the platform

Large text and contrast are baseline requirements, not a theme. Reading preferences, semantic structure, keyboard behavior, admin usability, and full web text all have to survive new content types and new features.

Public proof and operational proof are different

A source build can pass while a database migration, authenticated preview, scheduled job, or delivery integration still needs environment-specific verification. I learned to report each evidence layer separately instead of collapsing them into “it works.”

Current best

ArcadeProfile is both the public artifact and the product that ships the artifacts. Its current strength is the connection between an accessible reading experience and a publishing system with explicit content, audience, delivery, and audit boundaries.

Limits and open questions

- There are no public audience-growth or revenue figures to claim in this case study.
- Database, email, and authenticated-admin behavior require environment-specific verification in addition to a source build.
- The public site can fall back gracefully when CMS data is unavailable, but that is not proof that every connected service is healthy.
- A publication-ready screenshot of the public reading or projects surface still needs to be captured.